Performance Investigation of Adaptive Filter Algorithms and their Implementation for MIMO Systems

Jengis Lo Ming

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Engineering
in
Electrical and Electronic Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

2005

ABSTRACT

The Group Research department in Tait Electronics has a reconfigurable platform for MIMO research. In particular, the platform has an adaptive multivariate DFE with the LMS algorithm currently implemented. The LMS algorithm has been simulated and optimised for implementation on the FPGA. The main objective of the research is to investigate an alternative, the RLS algorithm by comparing its performance to the LMS algorithm. The RLS algorithm is known to be more complex than the LMS algorithm but offers the potential for improved performance due to its fast-converging nature. This thesis provides a performance investigation of these adaptive filter algorithms on the MIMO system for the purpose of real-time implementation on the Tait platform. In addition to performance investigation, stability analysis and a feasibility study is shown for the RLS algorithm on the FPGA. The research is industry based and is supported by FRST.

ACKNOWLEDGEMENTS

Firstly, I would like to thank Dr. Peter Smith for providing invaluable guidance and supervision during my final stage of research, especially proof reading my thesis. Special thanks to Peter Squires for introducing me to this industry-based project.

I would also like to thank Tait Electronics Ltd, namely Dr. Ian McLouglin, Howie Kuo and James Dowle for providing the opportunity to complete a research period in their Group Research department. I would also like to thank my colleagues over in Tait Electronics for those overly long coffee breaks.

In addition, I would like to thank FRST and the University of Canterbury for supporting me financially during my education in the form of a TIF Fellowship, University of Canterbury Masters Scholarship and Research Award.

Finally, I would like to thank my family and friends who have supported me through the difficult times while doing this research. God bless you all.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 GENERAL

MIMO (Multiple-Input Multiple-Output) is currently a promising area in wireless communication research due to its performance advantage over SISO (Single-Input Single-Output) and smart antenna systems. This advantage stems from the use of multiple antennas at the transmitter and receiver. This creates the possibility for multiple parallel channels to exist [3] and these channels can be used to increase data rate or improve performance. Great interest was stimulated by Foschini [4] and Telatar [3] in the 1990's who discovered the capacity benefits that are possible through the use of MIMO technology. Higher capacity means higher throughput and with 3^{rd} generation systems, the demand for higher data rates is constantly increasing. Hence, with research into 4^{th} generation systems already well under way, high rate communications is seen to be essential. MIMO is one technology which may provide these high rates, and in the last 15 years there has been an enormous research effort in this area, including information theory [5, 6], prototypes [7, 8], channel models [9, 10, 11] and channel measurements [12, 13, 14].

As part of this international research, the Group Research department in Tait Electronics has constructed a real-time reconfigurable platform for MIMO research. The platform currently has a 4 by 4 antenna arrangement, which is expandable to a 12 by 12 system by utilising three interlinked 4 channel processing modules. The module also contains mixed signal interfaces, RF filters and the adaptive multivariate DFE (Decision Feedback Equaliser). In particular, the Stratix FPGA (Field Programmable Gate Array) is used for the implementation of the adaptive multivariate DFE, which currently uses the LMS (Least Mean Squares) algorithm [2, 15]. The platform is designed to be software reconfigurable to support multiple algorithm development efforts.

The platform is an ideal environment for experimentation due to its reconfigurability. The reconfigurability of the platform has successfully provided a real-time channel sounder, space-time coded systems and the current multivariate DFE system [16, 17]. Furthermore, the platform allows changes in modulation schemes, the number of si-

multaneous transmitters and receivers, adaptive algorithms, etc. The equalisation procedure can be modified, for example to allow different training sequences, training lengths, decision-directed mode, etc. The adaptive multivariate DFE has been implemented on the platform since it is a well-known technique for combating interference in the unknown wireless channel. It is also a better alternative than the linear equaliser and the MLSE (Maximum Likelihood Sequence Estimator), because it performs better than the linear equaliser without the complexity of the MLSE [18].

A key component in communications is adaptive equalisation because it greatly improves signal quality, thus enabling reliable and efficient communication between users. Adaptive equalisation is an important signal processing technique for digital radio receivers, for example the Tait platform, and is an application of adaptive filter theory [19, 20, 21]. Basically, an adaptive equaliser changes its filter characteristic, namely its tap weights based on signal inputs that have passed through the unknown channel. In addition, adaptive equalisers are partly deployed for the mitigation of ISI (Inter Symbol Interference) and channel variation tracking. Adaptive equalisation is currently implemented on the Tait platform, via an adaptive multivariate DFE. The current adaptive algorithm on the DFE is the LMS algorithm. The LMS algorithm is a practical algorithm to implement due to its simplicity and comes from the family of stochastic gradient algorithms. In contrast, the RLS algorithm is an adaptive algorithm which can exhibit faster convergence, at the price of extra complexity and hardware requirements. The RLS algorithm is a special case of the Kalman filter. Due to advances in hardware design, the RLS algorithm is becoming a more popular possibility for implementation. Therefore, the performance of these algorithms will be considered via results from error curves, BER etc. There are two simulation scenarios in this thesis, a simple and a full system simulation. The simple simulation focusses on the performance of these algorithms on the multivariate DFE in general and uses the Jakes channel model. The full system simulation (or Tait simulation) uses specifications from the Tait platform and focusses on the feasibility of these algorithms for hardware implementation. The Tait simulation uses a quasi-stationary channel model, and real time captured data for part of the fixed point simulation. Furthermore, stability of these algorithms is analysed, in particular the stability of the RLS algorithm and its intermediate variables.

Although work on the LMS and RLS algorithms for SISO systems is extensive, the work here focusses on MIMO systems which is more recent. Literature regarding the RLS algorithm on a multivariate DFE is sparse, especially in the area of implementation and stability issues. Thus, the value of this thesis is in the analysis, performance investigation and eventually the feasibility assessment of these algorithms on a real-time system, where system features such as fixed point arithmetic, synchronisation, captured data, etc are all taken into account.

1.2 LITERATURE REVIEW

Due to the rapid development of the telecommunications industry in the last few decades, a large amount of literature can be found covering MIMO systems, numerous adaptive filters and various equaliser structures. In the field of adaptive equalisation, the leading contenders for implementation are typically the LMS and RLS algorithms, or their variants. Therefore, the thesis focussed on a performance investigation of the LMS and RLS algorithms on the multivariate DFE with the intention of implementing the RLS algorithm on the Tait platform.

The purpose of the DFE is to mitigate the ISI (Intersymbol Interference) which is prevalent in digital communication systems. The DFE was first proposed by Austin [22] and since then extensive literature has appeared on improvements to and variants of the initial DFE. The LMS and RLS adaptive algorithms can be applied to the DFE to create an adaptive DFE. This is useful for tracking a time-varying channel and is considered one of the most practical receiver structures in communication systems.

The LMS algorithm was first developed by Widrow and Hoff [23]. The RLS algorithm originated from the use of the Kalman filtering algorithm for estimation equaliser coefficients [24], and Falconer later recognised the Kalman filter to be a form of the RLS algorithm [25]. A number of variants of these two algorithms are described in [19, 21, 26, 27]. Most of the literature focusses on a performance comparison of the LMS and RLS algorithms. See for example the work in [28]. The origins of adaptive equalisation are described in detail in Qureshi's paper [29], providing an excellent review of the research in adaptive equalisation theory before 1985.

MIMO systems have been researched extensively because of their capacity advantage over SISO systems [30]. This has led to the development of the MIMO DFE [18, 31, 32, 33, 34, 35, 36] to increase the performance of the MIMO receiver. A number of publications have described and analysed different adaptation algorithms that have been implemented on the multivariate DFE. Generally, the literature has compared the LMS and RLS algorithms in different channel environments as seen in [37, 38, 39, 40, 41, 42]. However, some literature focussed on the RLS algorithm [43, 44] and its tracking performance using the Jakes channel model [45]. Fixed point effects of the RLS algorithm were also reviewed since hardware implementation is important [19, 46]. In addition, some real-time systems which applied the RLS algorithm were studied in [47]. This thesis focusses in the same area and aims to investigate the performance of these adaptive filters on the Tait platform, and subsequently assess their potential for implementation in hardware, notably the FPGA.

One of the key issues encountered during this thesis was the potential instabilities of the adaptive algorithms on the DFE, particularly the RLS algorithm. Therefore, the performance of the RLS algorithm on a SISO DFE was reviewed [48, 49, 50] to understand this behaviour and other work on instability was considered [51, 52, 53,

54, 55, 56]. In addition, decision delays on the DFE were studied [57, 58, 59, 60, 61] because of their effect in stabilising the $\mathbf{P}(k)$ matrix used in the RLS algorithm during one of the simulation scenarios. It was found that decision delays can help optimise DFE performance. In work that involves the RLS algorithm [62, 39, 40], decision delays have been discussed and shown to improve performance. However, these papers do not mention the behaviour of the $\mathbf{P}(k)$ matrix. The matrix $\mathbf{P}(k)$ is an intermediate variable in the RLS algorithm.

The stability of the RLS algorithm is essential since hardware implementation is considered. Since the use of optimal decision delays only help stabilise the $\mathbf{P}(k)$ matrix in one simulation scenario, another method was required to stabilise the RLS algorithm. Therefore, a divergence mitigation technique was discovered which resets the $\mathbf{P}(k)$ matrix. This technique originated in literature that did not involve adaptive equalisation [63, 64]. The reset technique was applied in adaptive controls and power applications, and surprisingly not mentioned in any recent communications or signal processing literature. Generally, references such as [65, 66, 67, 68, 19, 69, 50] provide a good overview of adaptive equalisation and its application to wireless communications.

1.3 SCOPE OF THE THESIS

The thesis is primarily focussed on the performance of adaptive filter algorithms and their implementation on MIMO systems. The LMS and RLS algorithms were selected for implementation on the multivariate DFE. Their performances and difficulties encountered were illustrated and discussed. Furthermore, implementation issues on hardware were also considered.

Chapter 2 of this thesis presents background information on MIMO systems. The Chapter briefly describes the MIMO channel and its capacity advantage over SISO channels. Chapter 3 covers the theory of adaptive equalisation. The SISO DFE is described and its adaptive extension. Similarly, the multivariate DFE is described as well as its adaptive extension via the normal equations. This is followed by a description of the LMS and RLS algorithms and their implementation on the multivariate DFE. Some difficulties relating to implementation are addressed. Chapter 4 describes the Tait platform, in particular its purpose, structure and the current algorithms implemented on it. In addition, the Chapter describes the platform's capability and measurement outputs as well as simulation scenarios. Chapter 5 presents the results from Matlab simulations focussing on a performance comparison between the LMS and RLS algorithms on the multivariate DFE. Difficulties and problems during the simulations are discussed and mitigation techniques suggested. The results show that the RLS algorithm outperforms the LMS algorithm when implemented on the multivariate DFE. During simulations, an undesirable phenomenon was encountered when adjusting the RLS parameter, λ, the forgetting factor. For values of $\lambda < 1$, the $\mathbf{P}(k)$ matrix diverged. The implementa-

tion of decision-delays on the multivariate DFE helped mitigate this problem. Chapter 6 presents the results from simulations using specifications from the Tait platform. The adaptive algorithms were simulated in two environments, floating and fixed point. In addition, real-time data extracted from the Tait platform were used in the fixed point simulation. This was extremely useful in evaluating the possibility of implementing the RLS algorithm on the Tait platform. During the Tait simulations, decision-delays did not help mitigate the diverging $\mathbf{P}(k)$ matrix. Therefore, an effective and novel method was discovered which resets the $\mathbf{P}(k)$ matrix. The method successfully stabilised the RLS algorithm. Chapter 7 describes the implementation issues involved in using the adaptive algorithms on the Tait platform. Resource requirements and a discussion regarding the feasibility of the adaptive algorithms are also given. Finally, Chapter 8 presents some conclusions and summarises the achievements and contributions of this thesis, as well as suggestions for future work.

1.4 CONTRIBUTIONS

The work in this thesis considers many issues regarding the implementation of the RLS algorithm on MIMO systems, in particular relating to its implementation on the multivariate DFE. The main contribution of this thesis is an assessment of any performance advantage of the RLS algorithm over the LMS algorithm in two different simulation scenarios; a simple simulation and a full system Tait simulation. The simple simulation, referred to in this thesis as the base simulation, removes the complications of the Tait platform to evaluate the performance of the adaptive algorithm on the multivariate DFE. The Tait simulation includes all specifications from the Tait platform, and similar to the base simulation, evaluates the performance of the adaptive algorithms in a floating and fixed point environment. Furthermore, some of the fixed point simulations use real-time captured data from the Tait platform.

During the performance investigation of the LMS and RLS algorithms on the multivariate DFE, instability was encountered in the RLS algorithm. Possible instability in fixed point is well-known, but this thesis discovered the same issue, during floating point as well as fixed point simulations. The source of the instability was a variable in the RLS algorithm, the inverse correlation matrix, $\mathbf{P}(k)$, which diverges and eventually causes instability. In addition, this variable is sensitive to the parameter, λ, the forgetting factor. Furthermore, this instability occurred for different simulation scenarios as mentioned above, which includes different channel models. However, an effective method was discovered, which mitigated the instability in the RLS algorithm by resetting the $\mathbf{P}(k)$ matrix. This approach solved the instability problem in the full Tait simulations with captured data, which is the closest simulation to reality.

One of the initial goals of this thesis was the real-time implementation of the RLS algorithm on the Tait platform. Generally, the results in the floating point simulations

showed that the RLS algorithm outperformed the LMS algorithm which is consistent with literature. However, in fixed point simulation using specifications from the Tait platform, the RLS does not outperform the LMS algorithm, and the tracking behaviour of the RLS algorithm is poor compared to the LMS algorithm. Furthermore, the RLS algorithm is too complex for implementation on the current FPGA and therefore cannot be considered as an alternative to the more robust and practical LMS algorithm.

Chapter 2

MULTIPLE INPUT MULTIPLE OUTPUT SYSTEM BACKGROUND

2.1 INTRODUCTION

MIMO is a promising area of research in wireless communications. In communications theory, MIMO refers to radio links with multiple antennas on the transmitter and receiver, unlike SISO which uses a single antenna on the transmitter and receiver. The enormous interest in this area has developed due to the higher capacity MIMO provides over SISO system without extending the radio spectral bandwidth. Therefore, MIMO systems are more bandwidth efficient compared to SISO systems. This is extremely important for the future of wireless communications, due to the great demand for increased quality of service and throughput, especially in mixed media 4^{th} generation systems. The following Sections will summarise important aspects about MIMO channels including basic theory, capacity and channel models.

2.2 SYSTEM MODEL

Consider a system with n transmit and m receive antennas, giving an $m \times n$ channel. Figure 2.1 shows a discrete-time model of this channel. Based on Figure 2.1, each individual channel connects one of the transmit antennas to a receive antenna, therefore the symbols are transmitted over $m \times n$ channels. The received signals $y_1(k)$ to $y_m(k)$ are mathematically defined below. The following equation represents the discrete output of the j^{th} receiving antenna, $y_j(k)$.

$$y_j(k) = \sum_{i=1}^{n} x_i(k) h_{j,i}(k) + v_j(k) \tag{2.1}$$

In (2.1), $x_i(k)$ is the discrete input to the i^{th} transmitting antenna and $h_{j,i}(k)$ is the discrete channel impulse response (or a random complex channel coefficient) from the i^{th} transmitting antenna to the j^{th} receiving antenna at time k. Finally, $v_j(k)$ is the additive white noise at the output of the j^{th} receiving antenna at time k. Furthermore,

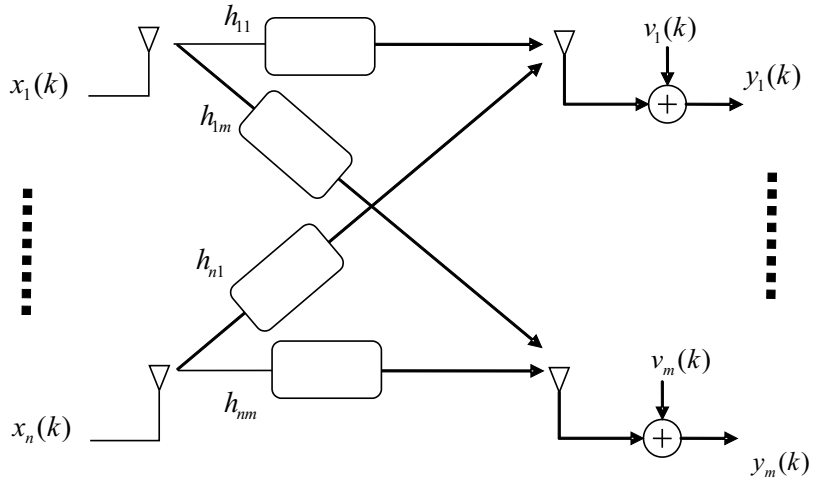

Figure 2.1 A n by m MIMO channel

the equation can be written in matrix form as below:

$$\mathbf{y}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{2.2}$$

In (2.2), $\mathbf{y}(k) = [y_1(k) \quad y_2(k) \quad \dots \quad y_m(k)]^T$, $\mathbf{x}(k) = [x_1(k) \quad x_2(k) \quad \dots \quad x_n(k)]^T$, $\mathbf{v}(k) = [v_1(k) \quad v_2(k) \quad \dots \quad v_m(k)]^T$ and $\mathbf{H}(k)$ is the channel matrix where $h_{i,j}(k)$ is its $(i,j)^{th}$ element as shown in (2.3).

$$\mathbf{H} = \begin{bmatrix} h_{1,1}(k) & \dots & h_{1,n}(k) \\ h_{2,1}(k) & \dots & h_{2,n}(k) \\ \vdots & \ddots & \vdots \\ h_{m,1}(k) & \dots & h_{m,n}(k) \end{bmatrix} \tag{2.3}$$

In this thesis, a mixture of channel models are used for the entries of the **H** matrix. These models include the Jakes model which approximates a time-varying Rayleigh fading channel. For the Tait simulations, two types of channels are considered. The first type is a quasi-stationary channel model. The second type is based on measured data. Actual received data from the Tait platform are used meaning that real-time received data are used for simulating the adaptive multivariate DFE.

2.3 CHANNEL MODELS

Wireless (or mobile) channels can be described as having two fundamental phenomena [1], fading and delay spread:

- Fading (or Doppler spread) is a term describing the variations of the amplitude or relative phase, or both, of one or more of the frequency components of the received signal. These variations are caused by changes in the characteristics of the propagation path with time. Flat fading is an example of fading in which all frequency components of a received signal vary in the same proportion simultaneously when signal bandwidth is small.

- Delay spread is a type of distortion due to multipath and reflections. Signals arrive at different speeds due to different paths resulting in different time delays. Delay spread is known to cause ISI which is undesirable in digital communication systems.

- Frequency selective fading is when both fading and delay spread appear together. It causes different frequencies of a transmitted signal to be attenuated and phase shifted differently in a channel. This give rise to real challenges in modulation and detection, especially if the fading is fast. The channel can be modelled as a time variant linear filter.

These two mobile channel phenomena, fading and delay spread, can coincide or be separate in any given scenario. Assuming a data signal with period T and a receiver using coherent detection with a one-symbol processing window, a diagram can be generated as shown in Figure 2.2. The Figure indicates the consequences of signalling rate on the model. The faster the transmission, the smaller the value of T, therefore the greater the signal bandwidth becomes. This in turn makes the channel more frequency selective over that bandwidth. On the other hand, the slower the transmission, the faster the channel relatively becomes, and issues like pulse distortion and ISI appear. Note that f_D is the Doppler shift and τ_d is the delay spread. The Doppler shift f_D can be calculated as shown in (2.4).

$$f_D = \frac{f_c}{c} v \tag{2.4}$$

where f_c is the carrier frequency, c is the speed of light and v is the vehicle speed. The delay spread τ_d can be obtained by measuring the range of delays shown by the impulse response of the channel. As we can see from Figure 2.2, the combination of these two phenomena (Doppler and Delay spread) and the symbol period, T, allows various scenarios. The term frequency selective and flat are opposites of each other and are dependent on the delay spread τ_d and bandwidth W of the signal. Frequency selective transmission is typically when the range of delays are significant enough to affect the signal by causing part of the signal to be notched out. On the other hand, flat transmission is when the delays are insignificant and does not cause variation of the frequency response across the signal band. The literature in [1] provides more detail on delay and Doppler spread.

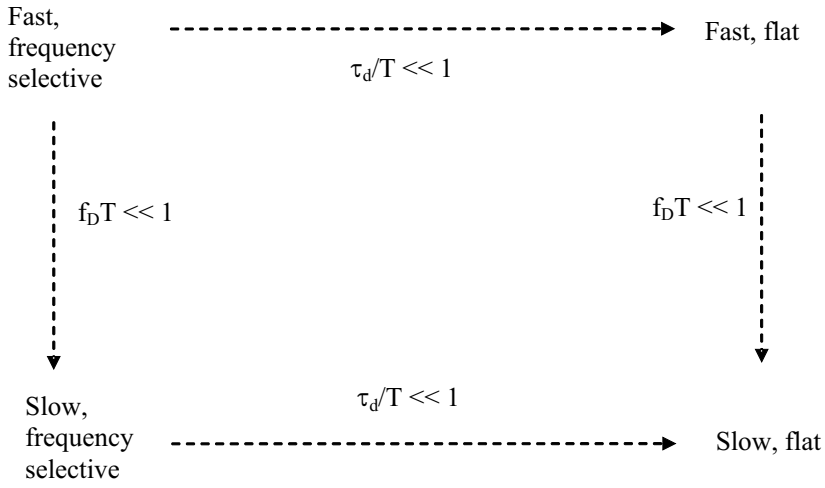

Figure 2.2 Diagram of various scenarios for a wireless channel [1]

2.4 CAPACITY

One important system measure in which SISO and MIMO channels behave very differently is capacity. Capacity is a measure of how much information can be transmitted and received with a negligible probability of error. In [70], it was shown that the multipath characteristic of radio transmissions can be used to multiplicatively increase the capacity of a radio system when MIMO is used. This is important for real-time applications since more data throughput means an increase in system capability.

In [69, 30], channel capacity is defined as in (2.5), where X and Y are random variables which represents the input and output of a memoryless wireless channel respectively. $I(X;Y)$ represents the *mutual information* between X and Y. Mutual information is a measure of the amount of information that one random variable contains about another variable.

$$C = \max_{p(x)} I(X;Y) \tag{2.5}$$

Equation (2.5) states that the mutual information is maximised with respect to all possible statistical distributions, with density $p(x)$, of the transmitted signals. The mutual information, $I(X;Y)$ is defined by:

$$I(X;Y) = H(Y) - H(Y|X) \tag{2.6}$$

where $H(Y)$ represents the entropy of Y and $H(Y|X)$ is the conditional entropy between the random variables X and Y. The entropy of a random variable can be described as a measure of the amount of information required on average to describe

the random variable. It can also be described as a measure of the uncertainty of the random variable. In addition, the mutual information between X and Y depends on the properties of the channel (through a channel matrix, $\mathbf{H}$ as seen in (2.3)) and the properties of X and Y (through their probability distributions). To summarise the capacity advantage of MIMO systems over SISO systems, results for their ergodic (mean) capacities are shown below.

For a SISO system, $m = n = 1$ representing one transmit and one receive antenna. The random complex channel coefficient of the SISO link is $h_{1,1}$. The mean capacity is given by (2.7) [30].

$$C = E_H\{log_2(1 + \rho.|h_{1,1}|^2)\} \tag{2.7}$$

where E_H denotes the expectation over all channel realisations and ρ represents the average signal-to-noise ratio (SNR). For the MIMO case, a similar matrix equation exists as shown in (2.8) [30].

$$C = E_H\big\{log_2\big[det\big(\mathbf{I}_m + \frac{\rho}{n}\mathbf{H}\mathbf{H}^H\big)\big]\big\} \tag{2.8}$$

In (2.8), ρ is the average signal-to-noise ratio (SNR) at each receiver branch and $\mathbf{H}$ is the channel matrix. In [30], the results for various SNR values show that MIMO outperforms SISO systems in terms of capacity. This performance advantage can be seen by a consideration of (2.7) and (2.8). In (2.7), the only parameter that can be altered is the SNR, ρ. Increasing ρ increases C logarithmically for both SISO and MIMO [3, 4]. For MIMO systems, m and n can also be altered and increasing both m and n increases MIMO capacity linearly, under certain constraints on $\mathbf{H}$ [3, 4]. Therefore, MIMO systems are more spectrally efficient compared to SISO systems. Another method for using multiple antennas in a communication system is the smart antenna system which utilises multiple antennas at the receiver only. However, [37] shows results for various numbers of receive antennas and the capacity for MIMO systems also outperforms the smart antenna array system. Evidently, from this brief description regarding the capacity advantage of the MIMO system, MIMO is a promising area of research showing potential increase in system capability.

2.5 OVERVIEW

An overview of MIMO systems has been given in this Chapter, providing descriptions of some of the models used and discussing the advantages of MIMO over SISO systems based on capacity. There is now an enormous literature including review articles [71, 72], tutorials [30, 73, 74] and text books [75, 76, 77] which describe MIMO systems. Hence, the overview of this Chapter is very brief. The main thrust of this thesis is an investigation of the LMS and RLS algorithms on a multivariate DFE. For this purpose,

further background in MIMO specifics is unneccessary.

Chapter 3

ADAPTIVE EQUALISATION THEORY

3.1 INTRODUCTION

Adaptive equalisers of various types are used extensively in communication systems. Tait Electronics are currently considering the use of more complex adaptive algorithms and are interested in their feasibility and performance compared to the existing system [2]. The purpose of communication channel equalisation is to mitigate the effects of ISI and channel variation. Combining adaptive filters with channel equalisers increases the robustness of wireless communication systems.

The current system at Tait Electronics employs a multivariate DFE, which uses a multiuser detector structure. The DFE is accepted as preferable to the linear and MLSE equalisers based on performance and complexity. This is because the DFE performs almost as well as the MLSE equaliser but with the lesser complexity of a linear equaliser [18]. In addition, even though linear equalisers are alternative solutions to the ISI problem, they have the disadvantage of noise enhancement as discussed in [50].

This Chapter will describe the theory of adaptive filters, equaliser structures and their implementation. A Section on adaptive filters and their basic theory is given first. This is followed by a Section on the SISO DFE and its adaptive version. The MIMO extension of the DFE and its adaptive version is then described and illustrated.

3.2 ADAPTIVE ALGORITHMS

3.2.1 Background

In this Section, a summary of the two adaptive algorithms are shown; LMS and RLS. Both algorithms are adaptive filter algorithms that differ in terms of performance and complexity as discussed in [19]. These adaptive algorithms are implemented on the multivariate DFE using a set of normal equations. In wireless communications, adaptive techniques in equalisation help compensate for the unknown channel.

3.2.2 LMS Algorithm

The LMS algorithm belongs to the family of stochastic gradient algorithms. The stochastic gradient approach uses a deterministic gradient in a recursive computation of the Wiener filter for stochastic inputs. The algorithm is simple to implement, requiring no matrix inversions or measurements of the correlation matrix. The LMS algorithm is a linear adaptive filtering algorithm that consists of two basic processes:

- Filtering Process: This process involves the computation of the output of the transversal filter and the generation of the error by comparing the desired and estimated output.
- Adaptive Process: This is automatic adjustment of the tap weights according to the error produced.

The LMS algorithm is comparable to another adaptive technique called the method of steepest descent which is frequently used in optimisation theory. It uses an iterative technique for the optimisation of a set of variables by minimising some cost function. Both adaptive techniques use the gradient vector, but in a different form. To explain the LMS algorithm, consider a transversal filter with L taps, thus having a $L \times 1$ tap-weight vector $\hat{\mathbf{w}}(k)$. At time k, the system has a desired (transmitted) signal $x(k)$ which is a scalar, and the $L \times 1$ tap-input (received) vector $\mathbf{y}(k)$. The error signal is $e(k)$ which is a scalar (see (3.6)). The gradient vector in the method of steepest descent requires the tap-input vector $\mathbf{y}(k)$ and error signal $e(k)$ as shown in (3.1).

$$\bigtriangledown(k) = -2E[e^*(k)\mathbf{y}(k)] \tag{3.1}$$

The gradient vector $\bigtriangledown(k)$ is the direction of the greatest rate of change which is described in detail in [19]. Basically, the method of steepest descent is described by (3.2) below.

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) - \mu\bigtriangledown(k) \tag{3.2}$$

where μ is a positive constant called the step-size parameter. Rather than using the expected value of the gradient vector as shown in (3.1), the LMS algorithm uses the instantaneous estimate of the gradient vector as shown in (3.3).

$$\hat{\bigtriangledown}(k) = -2e^*(k)\mathbf{y}(k) \tag{3.3}$$

Therefore, applying this instantaneous estimate to the method of steepest descent we get

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) - \mu\hat{\bigtriangledown}(k) \tag{3.4}$$

The LMS algorithm can then be formulated as shown below in (3.5)-(3.6)

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \mu \mathbf{y}(k) e^*(k) \tag{3.5}$$

where

$$e(k) = x(k) - \hat{\mathbf{w}}^H \mathbf{y}(k) \tag{3.6}$$

Note that $x(k)$ is the actual transmitted signal, superscript ^{H} is the Hermitian transpose and * is the complex conjugate. The weight vector $\hat{\mathbf{w}}(k)$ is initialised as a zero vector and the step-size parameter μ is carefully selected such that the LMS algorithm is stable.

3.2.3 RLS Algorithm

An alternative to the LMS algorithm is the RLS algorithm. Although more computationally demanding compared to the LMS algorithm, it can achieve better performance compared to the LMS algorithm because of its fast-converging property. The RLS algorithm is a special case of the Kalman filter, utilising the least squares approach to optimise the equaliser coefficients. The algorithm deals directly with the input data sequences and obtains estimates of correlations from the data.

The RLS algorithm exploits the method of least squares and a relation in matrix algebra known as the *matrix inversion lemma*. An important feature of the RLS algorithm is that it makes use of an exponentially weighted sum of squared errors extending back to the instant of time when the algorithm is initiated. The rate of convergence for RLS is typically an order of magnitude quicker than the standard LMS algorithm. A full derivation and description of the RLS algorithm can be seen in [19], which describes the cost function that is to be minimised and how the *matrix inversion lemma* is utilised to produce the inverse correlation matrix $\mathbf{P}(k)$.

To describe the RLS algorithm, consider a similar system to the LMS algorithm description, using a transversal filter with L taps. At time k, the filter has a $L \times 1$ tap-weight vector $\hat{\mathbf{w}}(k)$, desired (transmitted) signal $x(k)$ and the $L \times 1$ tap-input (received) vector $\mathbf{y}(k)$. The RLS algorithm calculates the tap weights such that $\hat{\mathbf{w}}^H(k)\mathbf{y}(k)$ is an estimate of $x(k)$. A summary of the RLS algorithm is shown below. For each instant of time $k = 1, 2, \ldots$, compute:

$$\mathbf{K}_p(k) = \frac{\lambda^{-1}\mathbf{P}(k)\mathbf{y}(k)}{1 + \lambda^{-1}\mathbf{y}^H\mathbf{P}(k)\mathbf{y}(k)} \tag{3.7}$$

$$e(k) = x(k) - \hat{\mathbf{w}}^H(k)\mathbf{y}(k) \tag{3.8}$$

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \mathbf{K}_p(k) e^*(k) \tag{3.9}$$

$$\mathbf{P}(k+1) = \lambda^{-1}\mathbf{P}(k) - \lambda^{-1}\mathbf{K}_p(k)\mathbf{y}^H(k)\mathbf{P}(k) \tag{3.10}$$

In (3.7)-(3.10), the parameter λ is the forgetting factor and the matrix $\mathbf{P}(k)$ is the inverse correlation matrix, these are discussed more fully below. The algorithm is initialised by setting $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$ where δ is a small positive constant, and $\hat{\mathbf{w}}(0) = \mathbf{0}$. Typically, the forgetting factor λ is selected to be close 1. The RLS algorithm can be implemented within a DFE simply by changing the input vectors to a set of normal input vectors consisting of concatenated inputs to the feedforward and feedback filters, which is introduced later in Section 3.3.2 and 3.4.4. There are two parameters in the RLS algorithm that are significant during the implementation and performance study. These are the forgetting factor λ and the inverse correlation matrix $\mathbf{P}(k)$.

The forgetting factor, λ ensures that data in the distant past are *forgotten* in order to afford the possibility of following the statistical variation of the observable data when the filter operates in a nonstationary environment. As mentioned previously, the value of λ is a positive constant, typically close to but less than 1. When $\lambda = 1$, it is basically the ordinary method of least squares and the algorithm has *infinite memory*. [19] describes this in more detail by introducing a weighting factor into the cost function that is to be minimised. The inverse correlation matrix $\mathbf{P}(k)$ is the inverse of the correlation matrix $\Phi(k)$ which can be derived using the *matrix inversion lemma* and is defined in (3.11). The correlation matrix is Hermitian, Toeplitz and almost always positive definite meaning that it is nonsingular allowing an inverse of this matrix to exist. From [19], the correlation matrix is seem to have a considerable impact on the performance of the adaptive algorithms. The matrix $\Phi(k)$ is defined by:

$$\Phi(k) = \sum_{i=1}^{N} \lambda^{N-i}\mathbf{y}(i)\mathbf{y}^H(i) \tag{3.11}$$

where $k = 1, \ldots, N$, and N is the length of the observed data sequence. The correlation matrix $\Phi(k)$ or the inverse correlation matrix $\mathbf{P}(k) = \Phi^{-1}(k)$ play a central role in the performance of the RLS algorithm. For example, in the standard convergence analysis of RLS [19] it is shown that the MSE of the weight vector is:

$$MSE = \frac{\sigma^2}{N - L - 1}\sum_{i=1}^{L}\frac{1}{\lambda_i} \tag{3.12}$$

where σ^2 is the variance of the errors, $e(k)$, L is the number of taps and N is the length of the observed data. The eigenvalues, λ_i, are the eigenvalues of $\Phi(k)$. Clearly, small eigenvalues increase the MSE and have a negative impact on the performance of the RLS algorithm. Another impact of small eigenvalues is the potential for "explosive divergence" in fixed point as discussed in [19, 78]. Finally since $\Phi(k)$ is a correlation matrix, both $\Phi(k)$ and $\mathbf{P}(k)$ should be Hermitian symmetric and positive definite.

Recursive computation of these matrices can lead to a loss of symmetry and positive definiteness, especially in fixed point. This loss of the correct matrix structure can also lead to explosive divergence [19]. Hence, problems encountered with the RLS algorithm may be caused by loss of symmetry or small eigenvalues. Finally, $\Phi(k)$ is often strongly diagonal [79] and this usually leads to improved performance. In summary, the ideal $\Phi(k)$ matrix is strongly diagonal, Hermitian, positive definite and has large eigenvalues. The ideal $\mathbf{P}(k)$ matrix is also Hermitian and positive definite and conversely has small eigenvalues.

3.2.4 Performance Survey

Based on literature in [19, 28, 80] the RLS algorithm generally outperforms the LMS algorithm in terms of error convergence rate. This in turn leads to a lower mean squared error (MSE) and bit error rate (BER). However, the performance of the RLS algorithm on the multivariate DFE is less widely known. Some work can be found in [43, 44, 37, 38, 39]. The literature in [19] also states that the LMS algorithm exhibits more robust tracking behaviour than the RLS algorithm in channel equalisation.

The issue in this thesis is the feasibility of the RLS algorithm in fixed-point precision and subsequently its possible implementation on the FPGA via the Tait platform. Furthermore, the adaptive algorithms are implemented on the multivariate DFE which may lead to complications during simulation and implementation. Performance investigation and feasibility of these adaptive algorithms are explored in this thesis.

3.3 SISO DFE

3.3.1 General SISO DFE

The key to understanding the MIMO DFE system is to study its SISO version. The SISO DFE consists of two tapped-delay-line filters, a feedforward and a feedback filter as shown in Figure 3.1 and a decision device. We define the transmitted signal as $x(k)$ and the received signal as $y(k)$. The filter output is $z(k)$ which goes through the decision device to produce an estimate $\hat{x}(k)$. Denote the length of the feedforward and feedback filters by L_{ff} and L_{fb} respectively. Therefore, the tap-weight vector for the feedforward filter is $\mathbf{w}^{ff}(k)$ of size $1 \times L_{ff}$ and for the feedback filter is $\mathbf{w}^{fb}(k)$ of size $1 \times L_{fb}$. The input to the feedforward filter is the $L_{ff} \times 1$ received vector $\mathbf{y}(k)$, consisting of $\big[y(k), \ldots, y(k - L_{ff} + 1)\big]^T$ and the input to the feedback filter is the $L_{fb} \times 1$ decision device output vector $\hat{\mathbf{x}}(k)$, consisting of $\big[\hat{x}(k), \ldots, \hat{x}(k - L_{fb} + 1)\big]^T$. Most commonly, DFEs are fractionally spaced meaning that the feedforward filter processes its input at the sampling rate, and the feedback filter processes its input at the symbol rate. Equation (3.13) shows the equation that describes the SISO DFE:

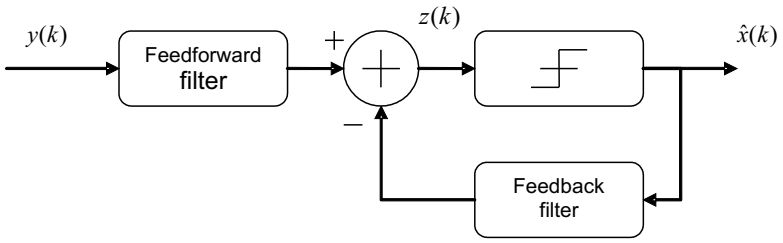

Figure 3.1 Simple block diagram for a SISO DFE

$$z(k) = \mathbf{w}^{ff}(k)\mathbf{y}(k) - \mathbf{w}^{fb}(k)\hat{\mathbf{x}}(k) \tag{3.13}$$

Note that the weight vectors are now defined as rows rather than columns, and vice versa for the input vectors. This change in notation is to match the standard definition in [65]. They can be easily reversed as seen later. The criteria for the DFE above is to minimise $E\big[|z(k) - x(k)|^2\big]$, which is known as the minimum mean squared error (MMSE) criterion. The weights in the feedforward and feedback filter are adjusted simultaneously to satisfy this criterion. More theory regarding this can be seen in [50]. The main purpose of the DFE is ISI suppression which is a form of interference in digitial communication systems namely in a fading radio channel. ISI distorts the received signal and the DFE counteracts this by suppressing it. The feedforward filter is designed to suppress the precursor ISI, and the feedback filter suppresses the postcursor ISI. Ideally, the complete equalised output signal should have only the desired result with noise assuming that the decision device outputs are correct.

3.3.2 Adaptive SISO DFE

The adaptive SISO DFE is well documented in [65, 50]. The results in [50] clearly illustrate the superiority of the RLS algorithm compared to the LMS algorithm and it also discusses the effects of the mobile channel, modulation schemes, practical limitations and complexity. However, it does not investigate the the extension of adaptive equalisation to MIMO systems, fixed-point analysis and hardware implementation, particularly on an FPGA. This thesis explores these issues in the context of the Tait platform [2]. The adaptive SISO DFE can be derived using the normal equations below and is described briefly in [65]. As before, we define $x(k)$ as the transmitted signal, $y(k)$ as the received signal and $z(k)$ as the filter output. A block diagram of an adaptive SISO DFE is shown in Figure 3.2. We also define the input vector to the feedforward filter, $\mathbf{y}(k)$ of size $L_{ff} \times 1$ and the input vector to the feedback filter, $\hat{\mathbf{x}}(k)$ of size $L_{fb} \times 1$

which are concatenated to produce $\mathbf{u}(k)$ of size $(L_{ff} + L_{fb}) \times 1$ as shown below:

$$\mathbf{u}(k) = \begin{bmatrix} \mathbf{y}(k) \\ -\hat{\mathbf{x}}(k) \end{bmatrix} \tag{3.14}$$

The basis of adaptive equalisation is a minimisation problem. The aim obviously is to minimise the error which is the difference between the desired and actual signal measured at each symbol. The error is used to determine the direction in which the tap-weights of the filter should be changed so as to approach an optimum set of values. With the given input vectors, we can simply reorganise the filtered output $z(k)$ as a product of two concatenated vectors as shown in (3.15). We take into account the possibility of the SISO DFE having more than 1 tap for each filter. Therefore, $\mathbf{w}^{ff}(k)$ and $\mathbf{w}^{fb}(k)$ are tap weight row vectors of length, L_{ff} and L_{fb} for the feedforward and feedback filters respectively.

$$z(k) = \begin{bmatrix} \mathbf{w}^{ff}(k) & \mathbf{w}^{fb}(k) \end{bmatrix} \begin{bmatrix} \mathbf{y}(k) \\ -\hat{\mathbf{x}}(k) \end{bmatrix} \tag{3.15}$$

Thus, with the previous definitions in this Section the minimisation problem must satisfy the criterion below:

$$\begin{bmatrix} \mathbf{w}^{ff}(k) & \mathbf{w}^{fb}(k) \end{bmatrix} = argmin \left[\left| \begin{bmatrix} \mathbf{w}^{ff}(k) & \mathbf{w}^{fb}(k) \end{bmatrix} \begin{bmatrix} \mathbf{y}(k) \\ -\hat{\mathbf{x}}(k) \end{bmatrix} - x(k) \right|^2 \right] \tag{3.16}$$

where $x(k)$ is the actual signal. As mentioned previously, the adaptive equalisation problem is a minimisation problem. The purpose is to find the optimal weights to minimise error. Adaptive algorithms do this in an iterative way.

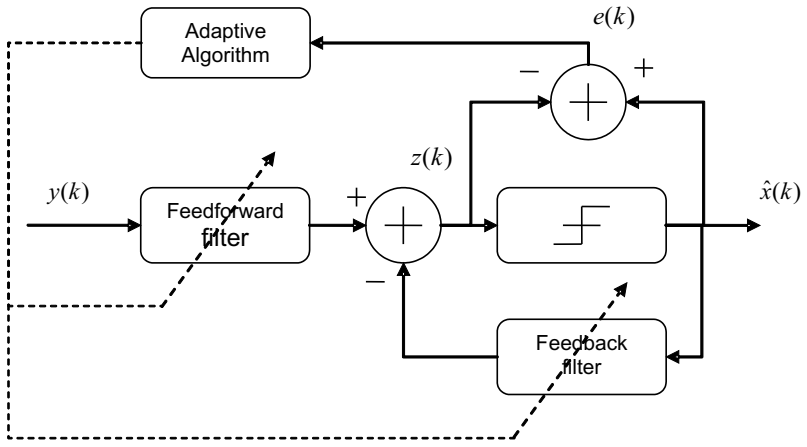

Figure 3.2 Simple block diagram for an adaptive SISO DFE

3.4 MIMO DFE

3.4.1 General MIMO DFE

The SISO DFE is widely studied, hence a more detailed description is not neccessary. Readers who are interested in the general theory should refer to the textbooks by Haykin [65, 19] or the literature review shown in Section 1.2. To design a MIMO DFE, we can simply extend the SISO DFE to its MIMO equivalent. There are two different design approaches as mentioned by [18, 34, 2].

- IR-DFE: Detect the signal from one user while treating the others as interference (Interference-Rejection approach)
- MUD-DFE: Detect the signals simultaneously (Multi-User Detection approach)

The current structure that is implemented on the Tait platform is the MUD-DFE. This approach performs much better than the interference rejection approach because it uses all the decision device output [2]. For ease of description of the multivariate DFE and its adaptive algorithm implementation, shown later in Section 3.4.2, we define the input vectors $\mathbf{y}(k)$ and $\hat{\mathbf{x}}(k)$ differently in terms of size and configuration since we now consider multiple antennas. This slight change in configuration will help readers understand how to implement these adaptive algorithms on the multivariate DFE and is a more direct match to the literature [38, 43, 44].

Consider a MIMO system where n is the number of transmit antennas and m is the number of receive antennas. We define $y_j(k)$ as the j^{th} received signal (i.e. the signal received on the j^{th} antenna), $\hat{x}_i(k)$ as the i^{th} signal after the decision device for the i^{th} DFE (i.e. the signal that estimates the transmitted signal from the i^{th} antenna) and $z_i(k)$ is the i^{th} signal before the decision device for the i^{th} DFE, where $j = 1, \ldots, m$ and $i = 1, \ldots, n$. Figure 3.3 shows an example of the MUD-DFE, showing the i^{th} DFE and its input signals to the i^{th} feedforward and feedback filters. Mathematically, Figure 3.3 is described by (3.17), where $z_i(k)$ can be written as:

$$z_i(k) = \sum_{\alpha=1}^{m} \mathbf{y}_\alpha(k)\mathbf{w}_{\alpha,i}^{ff}(k) - \sum_{\beta=1}^{n} \hat{\mathbf{x}}_\beta(k)\mathbf{w}_{\beta,i}^{fb}(k) \tag{3.17}$$

The inputs for all m feedforward and n feedback filters are the same as seen in (3.17). Referring to Figure 3.3 these inputs can be defined as $y_1(k), \ldots y_m(k)$ and $\hat{x}_1(k), \ldots, \hat{x}_n(k)$. Furthermore, each feedforward filter, as shown in Figure 3.3, has m subfilters and each feedback filter has n subfilters. Figure 3.4 shows an example of a 2×2 multivariate DFE. From (3.17), there are $n \times m$ feedforward subfilters and $n \times n$ feedback subfilters denoted by $\mathbf{w}_{\alpha,i}^{ff}$ and $\mathbf{w}_{\beta,i}^{fb}$ respectively, where $\alpha = 1, \ldots, m$ and $\beta = 1, \ldots, n$.

In Equation (3.17), there are 4 types of vectors, the feedforward weight vectors $\mathbf{w}_{\alpha,i}^{ff}(k)$, the feedback weight vectors $\mathbf{w}_{\beta,i}^{fb}(k)$, the input vector to the feedforward filter, $\mathbf{y}_\alpha(k)$ and the input vector to the feedback filter, $\hat{\mathbf{x}}_\beta(k)$. Assuming the length of the feedforward filter is L_{ff} and the feedback filter is L_{fb}, $\mathbf{y}_\alpha(k)$ has size $1 \times L_{ff}$, $\hat{\mathbf{x}}_\beta(k)$ has size $1 \times L_{fb}$. Therefore, $\mathbf{w}_{\alpha,i}^{ff}(k)$ is $L_{ff} \times 1$ and $\mathbf{w}_{\beta,i}^{fb}(k)$ is $L_{fb} \times 1$. A precise description of $\mathbf{y}_\alpha(k)$ and $\hat{\mathbf{x}}_\beta(k)$ is given in Section 3.4.2.

From (3.17) and Figure 3.3 it can be seen that there are n decision devices corresponding to the n transmitted signals. This also equates to a total of n feedforward and feedback filters.

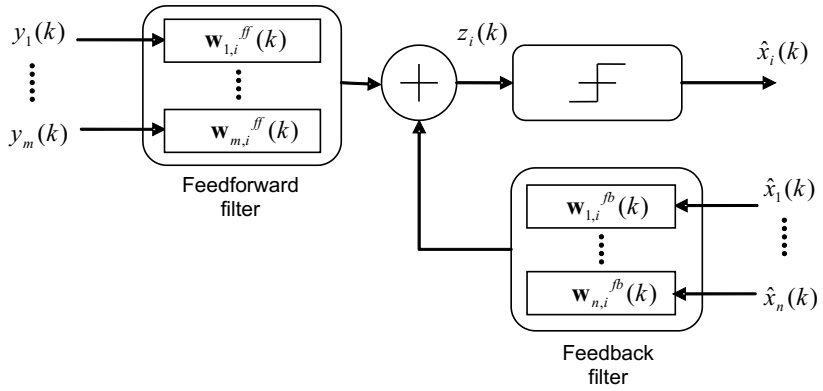

Figure 3.3 The i^{th} DFE for a multivariate DFE using multiuser detection where $i = 1, \ldots, n$

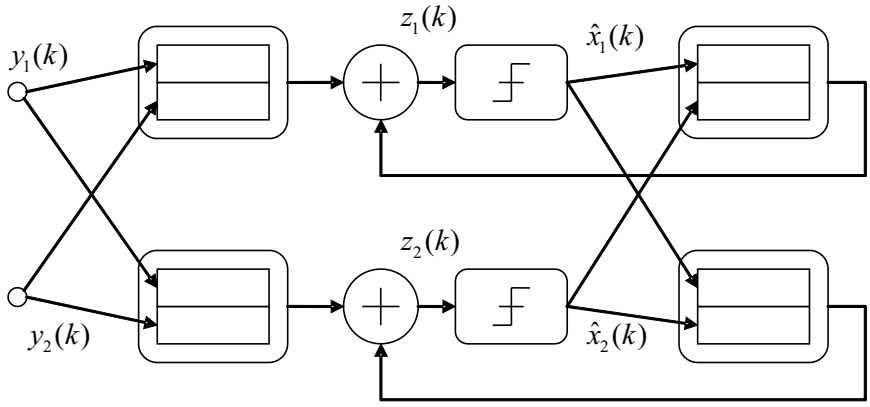

Figure 3.4 An example of a 2 by 2 multivariate DFE using multiuser detection

3.4.2 Adaptive Multivariate DFE

The concept of an adaptive DFE can be extended to the multivariate DFE by following the same approach for the SISO DFE using normal equations as seen in Section 3.3.2. The normal equations can be generated by concatenating the input vectors and weight vectors as shown later. The normal equations allow the implementation of any adaptive algorithm on the DFE to solve the minimisation problem. In detail, the input vectors for the normal equations are formed from the concatenation of the received vector $\mathbf{y}(k)$ and the negative of the output vector of the decision device, $-\hat{\mathbf{x}}(k)$, which results in $[\mathbf{y}(k) \quad -\hat{\mathbf{x}}(k)]$. Following the same approach for the SISO DFE, we can generate a minimisation problem statement for the adaptive MIMO DFE. To implement the MIMO DFE, we refer to (3.17) which describes the signal $z_i(k)$. Equation (3.17) can be further expanded for adaptive filter algorithm implementation by revealing the concatenated input vectors $\mathbf{y}(k)$ and $\hat{\mathbf{x}}(k)$:

$$\begin{aligned} z_i(k) &= \sum_{\alpha=1}^{m} \mathbf{y}_\alpha(k)\mathbf{w}_{\alpha,i}^{ff}(k) - \sum_{\beta=1}^{n} \hat{\mathbf{x}}_\beta(k)\mathbf{w}_{\beta,i}^{fb}(k) \\ &= [\mathbf{y}_1(k) \dots \mathbf{y_m}(k)| - \hat{\mathbf{x}}_1(k) \dots - \hat{\mathbf{x}}_n(k)] \begin{bmatrix} \mathbf{w}_{1,i}^{ff}(k) \\ \vdots \\ \mathbf{w}_{m,i}^{ff}(k) \\ \hline \mathbf{w}_{1,i}^{fb}(k) \\ \vdots \\ \mathbf{w}_{n,i}^{fb}(k) \end{bmatrix} \\ &= [\mathbf{y}(k)| - \hat{\mathbf{x}}(k)] \begin{bmatrix} \mathbf{w}_i^{ff}(k) \\ \hline \mathbf{w}_i^{fb}(k) \end{bmatrix} \end{aligned} \tag{3.18}$$

Note that the vectors $\mathbf{y}_\alpha(k)$, $\hat{\mathbf{x}}_\beta(k)$, $\mathbf{w}_{\alpha,i}^{ff}(k)$ and $\mathbf{w}_{\beta,i}^{fb}(k)$ have been defined in Section 3.4.1. Concatenated versions of these vectors are defined by (3.18) and have dimensions; $\mathbf{y}(k) : 1 \times mL_{ff}$, $\hat{\mathbf{x}}(k) : 1 \times nL_{fb}$, $\mathbf{w}_i^{ff}(k) : mL_{ff} \times 1$, $\mathbf{w}_i^{fb}(k) : nL_{fb} \times 1$. From (3.18), we can define a multivariate DFE version of the problem statement to calculate the filter weights, $\mathbf{w}_i^{ff}(k)$ and $\mathbf{w}_i^{fb}(k)$, such that the decision error $e_i(k)$ is minimised, where $e_i(k) = x_i(k) - z_i(k)$. The solution is:

$$\begin{bmatrix} \mathbf{w}_i^{ff}(k) \\ \hline \mathbf{w}_i^{fb}(k) \end{bmatrix} = argmin \left[\left| [\mathbf{y}(k)| - \hat{\mathbf{x}}(k)] \begin{bmatrix} \mathbf{w}_i^{ff}(k) \\ \hline \mathbf{w}_i^{fb}(k) \end{bmatrix} - x_i(k) \right|^2 \right] \tag{3.19}$$

From (3.18) above, we can define the entries of the input vectors. Firsly, we assume that $[y_j(1) \quad y_j(2) \quad \dots \quad y_j(N)]$ depicts a block of N samples from the output of the j^{th} receive antenna and $[\hat{x}_i(1) \quad \hat{x}_i(2) \quad \dots \quad \hat{x}_i(N)]$ depicts a block of N samples from the i^{th} decision-device, where $j = 1, 2, \dots, m$ and $i = 1, 2, \dots, n$. Once again, we

will assume that the length of the feedforward filter is L_{ff} and feedback filter is L_{fb}. Therefore, we define two matrices, $\mathbf{Y}_j : N \times L_{ff}$ and $\hat{\mathbf{X}}_i : N \times L_{fb}$ as shown in (3.20) and (3.21).

$$\mathbf{Y}_j = \begin{pmatrix} y_j(1) & 0 & \dots & 0 \\ y_j(2) & y_j(1) & \dots & 0 \\ \vdots & & \vdots & \\ y_j(L_{ff}) & \dots & \dots & y_j(1) \\ \vdots & \ddots & & \ddots \\ y_j(N) & \dots & \dots & y_j(N - L_{ff} + 1) \end{pmatrix} \tag{3.20}$$

$$\hat{\mathbf{X}}_i = \begin{pmatrix} \hat{x}_i(1) & 0 & \dots & 0 \\ \hat{x}_i(2) & \hat{x}_i(1) & \dots & 0 \\ \vdots & & & \vdots \\ \hat{x}_i(L_{fb}) & \dots & \dots & \hat{x}_i(1) \\ \vdots & \ddots & & \ddots \\ \hat{x}_i(N) & \dots & \dots & \hat{x}_i(N - L_{fb} + 1) \end{pmatrix} \tag{3.21}$$

From these input matrices, we can form the $N \times mL_{ff}$ $\mathbf{Y}$ matrix as shown below:

$$\mathbf{Y} = [\ \mathbf{Y}_1 \ \mid \ \dots \ \dots \ \mid \ \mathbf{Y}_m\ ] \tag{3.22}$$

From (3.22), we generate the input vector $\mathbf{y}(k)$, which is one row of the matrix $\mathbf{Y}$ at time k. Hence, $\mathbf{y}(k)$ is the k^{th} row of $\mathbf{Y}$ and is defined by:

$$\mathbf{y}(k) = [\ \mathbf{y}_1(k) \ \mid \ \dots \ \dots \ \mid \ \mathbf{y}_m(k)\ ] \tag{3.23}$$

The vectors $\mathbf{y}_j(k)$ are shown below where $j = 1, 2, \dots, m$:

$$\mathbf{y}_i(k) = [\ y_i(k) \ \dots \ y_i(k - L_{ff} + 1)\ ] \tag{3.24}$$

A similar approach to the above is used to generate $\hat{\mathbf{x}}(k)$ simply by replacing $\mathbf{Y}$ with $\hat{\mathbf{X}}$, constructed from $\hat{\mathbf{X}}_1, \dots, \hat{\mathbf{X}}_n$ and the length L_{ff} is replaced by L_{fb}. With the above definitions, we can generate $[\ \mathbf{y}(k) \ -\hat{\mathbf{x}}(k)\ ]$ for each time index k. This normal vector can then be applied to adaptive algorithms, namely the LMS and RLS algorithms as described later. The size of this concatenated input vector is $1 \times (nL_{ff} + mL_{fb})$. Figure 3.5 illustrates the i^{th} DFE for an adaptive multivariate DFE. Figure 3.6 shows an example of a 2×2 adaptive multivariate DFE. It is similar to Figure 3.4 except tap weights for the feedforward and feedback filters are modified based on the error signals.

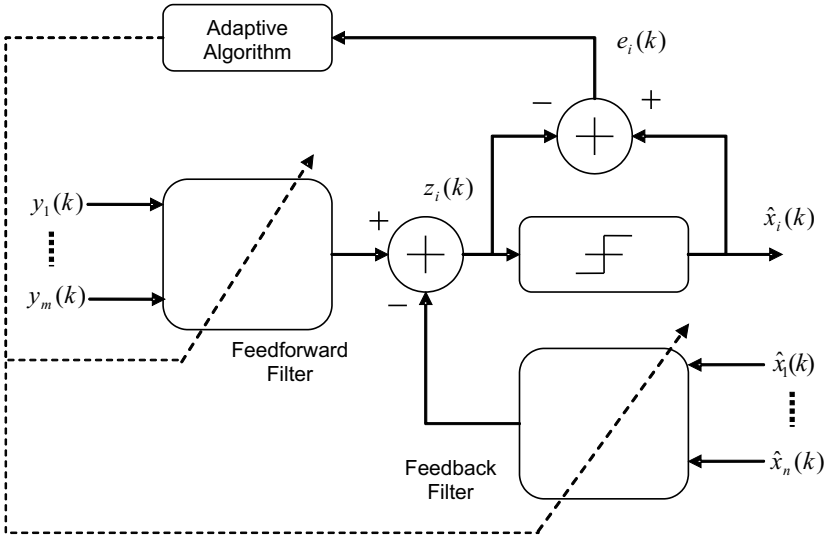

Figure 3.5 The i^{th} DFE for an adaptive multivariate DFE

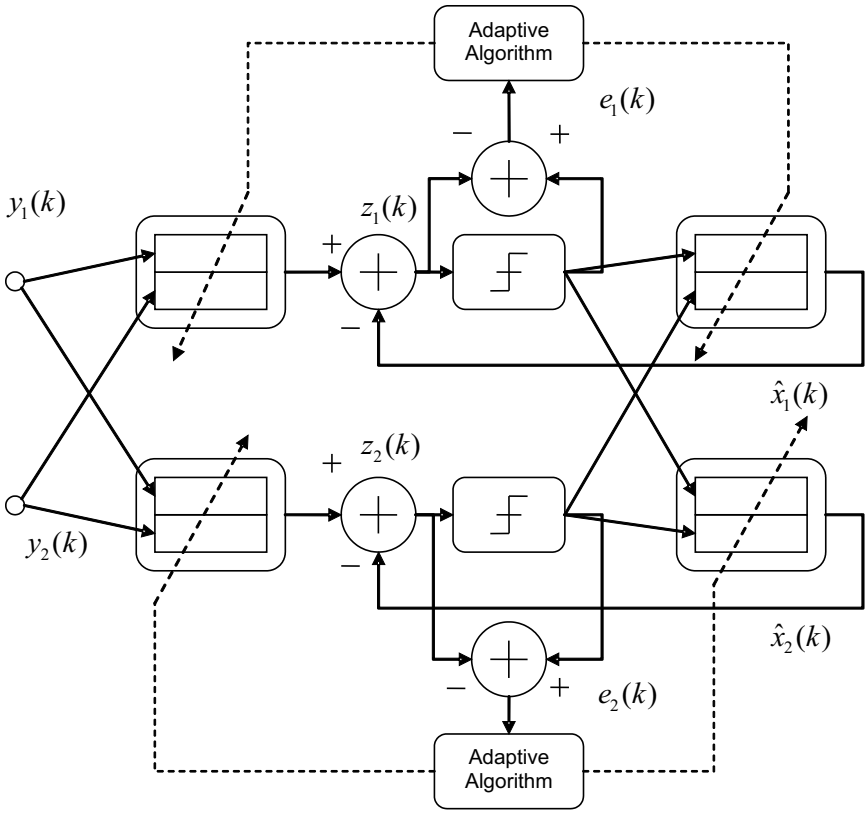

Figure 3.6 An example of a 2 by 2 adaptive multivariate DFE

3.4.3 LMS Algorithm Implementation

The LMS algorithm for the multivariate DFE is straightforward to implement. A detailed derivation of the LMS algorithm on the multivariate DFE can be seen in [38], which is similar to the LMS algorithm theory shown in Section 3.2.2 requiring the minimisation of the cost function seen in (3.19). Using the basis of the normal equations as shown previously, we can generate an LMS update algorithm during training mode as shown below:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \mu e_i(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H \tag{3.25}$$

The error signal for the i^{th} DFE is calculated by:

$$e_i(k) = x_i(k) - \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{w}_i(k) \tag{3.26}$$

where $i = 1, \ldots, n$, and the weight vector $\mathbf{w}_i(k) = [\mathbf{w}_i^{ff}(k)^T | \mathbf{w}_i^{fb}(k)^T]^T$ corresponds to the concatenated weights in the feedforward and feedback filters. As before, $x_i(k)$ is the signal sent from the i^{th} antenna. To produce the same update equation in decision-directed mode, simply replace $x_i(k)$ with $\hat{x}_i(k)$. The step-size parameter μ must be carefully selected to ensure stability in terms of error convergence. The typical approach for selecting this variable is by trial and error. In addition, the initial value of $\mathbf{w}_i(0)$ is usually chosen as the zero vector, $\mathbf{0}_{(mL_{ff}+nL_{fb})}$.

3.4.4 RLS Algorithm Implementation

The RLS algorithm implementation on the multivariate DFE has a similar approach to the LMS algorithm. Both require a cost function that is to be mininised, as seen in (3.19), by minimising the squared error for each data stream. The work in [44] describes how the RLS algorithm is implemented and [38] describes the derivation in more detail. To implement the RLS algorithm, the input vectors must be defined carefully because they play an important role in the algorithm. As mentioned previously, the input vectors are determined by the concatenation of the two inputs to the feedforward and feedback filters, $\mathbf{y}(k)$ and $\hat{\mathbf{x}}(k)$ respectively. With the given input vector configuration description, we can now formulate a RLS algorithm implementation on the multivariate DFE. As mentioned previously, the normal equations play a critical part in this implementation using $[\ \mathbf{y}(k) \quad -\hat{\mathbf{x}}(k)\ ]$. The weight update equation in training mode is given below as:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \mathbf{K}_p(k) \left(x_i(k) - \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{w}_i(k) \right) \tag{3.27}$$

where $x_i(k)$ is the signal sent from the i^{th} antenna. This is also known as the training sequence.

$$\mathbf{K}_p(k) = \mathbf{P}(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H R_e^{-1}(k)\lambda^{-1} \tag{3.28}$$

$$R_e(k) = \left(\begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{P}(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H + 1 \right) \lambda^{-1} \tag{3.29}$$

$$\mathbf{P}(k+1) = \left(\mathbf{P}(k) - \mathbf{K}_p(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{P}(k) \right) \lambda^{-1} \tag{3.30}$$

To begin the recursive algorithm, we initialise the variables as follows, $\mathbf{P}(0) = \mathbf{I}_{(mL_{ff}+nL_{fb})}$ and $\mathbf{w}_i(0) = \mathbf{0}_{(mL_{ff}+nL_{fb})}$. The forgetting factor λ is typically 1 or close to 1. The weight vector $\mathbf{w}_j(k)$ and gain vector $\mathbf{K}_p(k)$ are column vectors of length $mL_{ff} + nL_{fb}$. To produce the same update equation in decision-directed mode, simply replace $x_i(k)$ with $\hat{x}_i(k)$.

3.5 ISSUES

3.5.1 Introduction

The LMS algorithm is already succesfully implemented on the Tait platform without causing any major issues, for example instability. As long as the step-size parameter μ is selected carefully, the error curves produced by the LMS algorithm converge. However, this is not the case for the RLS algorithm implementation. Based on an initial study, this Section will describe various issues concerning the RLS algorithm relating to its instability. In addition, issues like tap lengths, decision-delays and tracking are also mentioned. Note that the effects of decision-delay and tap lengths affect the performance of both algorithms.

3.5.2 Initial RLS Algorithm Study

The initial study of the RLS algorithm implementation on the multivariate DFE used the structure outlined in [2]. It was found that the form of the RLS algorithm shown in [2] was inherently unstable. This version of the RLS algorithm originated from the literature shown in [43, 44]. This form of the RLS algorithm is shown below:

$$\mathbf{w}_i(k+1) = \lambda^{-1/2} \left(\mathbf{w}_i(k) + \mathbf{K}_p(k) \left(x_i(k) - \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{w}_i(k) \right) \right) \tag{3.31}$$

$$\mathbf{K}_p(k) = \mathbf{P}(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H R_e^{-1}(k) \tag{3.32}$$

$$R_e(k) = \left(\begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{P}(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H + 1 \right) \tag{3.33}$$

$$\mathbf{P}(k+1) = \left(\mathbf{P}(k) - \mathbf{K}_p(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{P}(k) \right) \tag{3.34}$$

Instability was immediately observed when modifying the forgetting factor λ using the equations of the form shown above. Based on the simulations, reducing the forgetting factor below 1 immediately causes divergence in the error curve. This meant that the RLS algorithm that was initially proposed only worked with $\lambda = 1$. This is somewhat desirable since implementation of this RLS algorithm requires no multipliers for the forgetting factor, since $\lambda = 1$ is equivalent to not having any multipliers at all. However, with $\lambda = 1$ the performance when tracking a changing channel is expected to deteriorate. As we know, the purpose of adaptive algorithms like the RLS is to converge towards optimal weights by slowly *learning* them via adaptation. Once the *learning* of weights reaches near completion, the algorithm reaches steady-state mode and continues to track the channel. Unfortunately, for this particular form of the RLS algorithm, the weights will never reach a steady state mode if $\lambda < 1$ because it will always diverge. The problem was located in the weight update equation having λ as an overall scalar multiplier shown as $\lambda^{-1/2}$ in (3.31). The mathematical explanation regarding this divergence is shown in (3.36) below.

$$\begin{aligned} \mathbf{w}_i(k+1) &= \lambda^{-1/2}[\mathbf{w}_i(k) + \mathbf{K}_p(k)(x_i(k) - [\ \mathbf{y}(k) \quad \tilde{\mathbf{x}}(k)\ ]\mathbf{w}_i(k))] \\ &= \lambda^{-1/2}[\mathbf{w}_i(k) + \mathbf{K}_p(k)e_i(k)] \end{aligned} \tag{3.35}$$

In steady state as the error signal $e_i(k)$ approaches zero we have:

$$\begin{aligned} \mathbf{w}_i(k+1) &= \lim_{e_i(k)\to 0} \lambda^{-1/2}[\mathbf{w}_i(k) + \mathbf{K}_p(k)e(k)] \\ &= \lambda^{-1/2}\mathbf{w}_i(k) \end{aligned} \tag{3.36}$$

By observing (3.36), we can see that λ does not adjust the correlation matrix directly but is a scalar multiplier which inflates $\mathbf{w}_i(k)$ for any $\lambda < 1$. Therefore, the weights are never converging but only diverging. The only solution to avoid divergence is to set $\lambda = 1$, which is a limitation in terms of performance investigation. It is possible that this problem is a typographical error in the paper, since some alterations to the equations do give an RLS structure which works.

The inevitable instability and limitations of this form of RLS algorithm led to its modification. The equation saw some changes in the placement of the forgetting factor similar to that shown in [38, 39, 19]. The modification removes λ from the weight update equation and places it as shown in Section 3.4.4.

3.5.3 Filter Tap Length

The DFE consists of two filters, the feedforward and feedback. From base simulations shown in Chapter 5, increasing the order of these filters decreases its performance. This is a fundamental issue in implementation especially for the RLS algorithm, since filter orders partly determine complexity, for example the size of the $\mathbf{P}(k)$ matrix depends on the length of the feedforward filter L_{ff} and feedback filter L_{fb}. Most work relating to the RLS algorithm on a MIMO DFE are simulation based, therefore filter tap lengths are never an issue. However implementation on hardware requires tap lengths to be limited to a reasonable number such that the complexity and resource consumption is not excessive while producing the desired results. Hence, a small number of taps is preferred.

Based on the literature in [50, 28, 49], the results show that there are optimal tap lengths for the DFE filters to maximise performance. For numerical examples, [62] limited its tap lengths to 7 feedforward and 1 feedback taps. In [49], it was shown that a DFE with 4 feedforward and 4 feedback taps outperformed the one with 12 feedforward and 20 feedback taps. Note these tap lengths examples coincide with a RLS algorithm implementation on a SISO DFE.

Other work involving decision delays without the RLS algorithm implementation, for example in [28] show that the number of taps is related to the decision delay Δ and channel memory v. Other literature like [60] developed fast algorithms for calculating delays to optimise finite-length DFEs. However, trial-and-error methods are usually sufficient to gauge reasonable values of the number of tap weights and the length of the decision delay. In addition, the LMS algorithm implementation on the Tait platform uses 4 feedforward and feedback taps. Hence 4 is the most convenient number of taps to use for the RLS algorithm implementation. Also, the full system simulation shown in Chapter 6 illustrates that tap lengths greater than 2 are required. Therefore, 4 is adopted as the standard in this thesis.

3.5.4 Effect of Decision Delay

In Section 3.4.3, the normal equations based on [2, 43, 44] used no decision delay, i.e. the weights operate on the vector containing $\mathbf{y}(k)$ and $-\hat{\mathbf{x}}(k)$. In most of the literature [19, 65, 18, 39] decision delays are used and so the weights are applied to the vector containing $\mathbf{y}(k)$ and $-\hat{\mathbf{x}}(k-\Delta)$, where Δ is the decision delay. From the base simulations shown in Chapter 5, we observe that without delayed decisions into the feedback filters in the multivariate DFE, $\mathbf{P}(k)$ exhibits divergence. By implementing delayed decisions the $\mathbf{P}(k)$ matrix converged. The general conclusion is that the delays affected the property of $\mathbf{P}(k)$. Analysis in SISO simulations showed that the delays removed some large covariance terms between the feedforward and feedback input vectors. The only scenario when $\mathbf{P}(k)$ showed convergence without using delayed decisions is when

the tap length is set to 1. The effect of adding decision delays for tap lengths greater than 1 is related to the property of the $\mathbf{P}(k)$ matrix. The delays can help maintain its strong diagonal structure. This is explained in detail in Section 3.6.3.

In most literature, as shown in [19, 65, 18, 39], delayed decisions were used in their DFEs whether it be SISO or MIMO. The major advantage of implementing delayed decisions using the RLS algorithm is a lower MSE as discussed in [62]. Therefore, decision delays can help improve the MSE and maintain convergence in the $\mathbf{P}(k)$ matrix. Optimum values for the decision delays can be calculated based on channel memory and filter tap lengths [81]. However, channel memory is usually unknown for real-time systems and therefore requires simulations or measurements.

In summary, the focus was not on how delayed decisions improves performance but on how they affect $\mathbf{P}(k)$. Evidently, this phenomenon was not encountered in the LMS algorithm because the algorithm only consists of a weight update equation and no intermediate values like $\mathbf{K}_p(k)$ and R_e.

3.6 INSTABILITY

3.6.1 Introduction

Instability of the adaptive algorithms can be determined by observing the error curves. The error curves in this thesis are defined as plots of the absolute magnitude of the error signal, $|e_i(k)|$ against time, where $e(k)$ is defined in (3.26). This Section describes the main causes of instability with the adaptive algorithms with more focus on the RLS algorithm. From the previous Section, the RLS algorithm exhibited instability with the form of equation shown in [2]. The RLS algorithm can also become unstable if the $\mathbf{P}(k)$ matrix is diverging due to fixed point effects in hardware. The saturation effect of fixed-point arithmetic causes some values in $\mathbf{P}(k)$ to be truncated to a certain maximum value which changes the property of the matrix.

3.6.2 LMS Algorithm

The LMS algorithm hardly experiences any instability because it only consists of a simple weight update equation. Without including the weight and input vectors, the LMS algorithm is dependent only on the step-size parameter, μ. The selection of μ is essential for determining reasonable error convergence rate while maintaining stability.

3.6.3 Inverse Correlation Matrix

The inverse correlation matrix, $\mathbf{P}(k)$, is an integral part of the RLS algorithm. During the analysis of the RLS algorithm, $\mathbf{P}(k)$ exhibited undesirable behaviour, even for a simple simulation of a static channel. During adaptation in training mode, the

RLS algorithm exhibited an almost exponential divergence in the size of the largest entry in the $\mathbf{P}(k)$ matrix. Observation showed that each variable converged except the $\mathbf{P}(k)$ matrix which kept increasing as the forgetting factor was reduced. The only scenario where the RLS algorithm did not show divergence is when λ is 1. This is an instability problem in two ways. Firstly, $\mathbf{P}(k)$ is not converging and secondly, in hardware implementation the saturation effect in fixed-point arithmetic will saturate the value of $\mathbf{P}(k)$ which in turn can modify the properties of the matrix.

$$\mathbf{P}(k+1) = \Big(\mathbf{P}(k) - \mathbf{K}_p(k) \left[\begin{array}{cc} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{array} \right] \mathbf{P}(k)\Big)\lambda^{-1} \tag{3.37}$$

The problem was tracked down to the Riccati equation (3.37) and the eigenvalues of the correlation matrix. For each iteration, the value of $\Delta\mathbf{P}(k) = \mathbf{P}(k+1) - \mathbf{P}(k)$ keeps increasing. It is interesting that such problems, discussed in Section 3.2.3, are well known in fixed point [19] but not usually discussed for floating point calculations. For the base simulations, the problem was identified as being a result of small eigenvalues in $\Phi(k)$ and a loss of the strong diagonal structure. This is described further in Section 5.2.8. For the Tait simulations, the causes of any divergence are more difficult to identify, due to complications such as the extra modules in the Tait simulations and the use of captured data from the unknown channel. Whatever the explanation is, the RLS algorithm is limited to having $\lambda = 1$ to achieve a converging $\mathbf{P}(k)$ matrix. This means the RLS algorithm has infinite memory, which may not be suitable if the channel is changing quickly. One possible technique to mitigate this divergence is by resetting the $\mathbf{P}(k)$ matrix to its initial value, the identity matrix $\mathbf{I}$. As mentioned before, $\mathbf{P}(k)$ exhibited a strong diagonal structure during the simulations as well as using a diagonal matrix as an initial value. Thus, resetting the matrix to its initial value preserves the symmetry and highly diagonal property. This is a method used in adaptive control systems to prevent the divergence of the $\mathbf{P}(k)$ matrix [63, 64].

It is also observed that the multivariate DFE structure contributes to the $\mathbf{P}(k)$ divergence. In the base simulations shown in Chapter 5, the $\mathbf{P}(k)$ matrix converged when a suitable decision-delay is implemented. The problem was initially thought to be the absence of the decision-delay which was causing the $\mathbf{P}(k)$ matrix to be diverging for λ less than 1. However, Tait simulations shown in Chapter 6 with a decision delay did not mitigate this and therefore the solutions were reduced to either setting λ to 1 or resetting the $\mathbf{P}(k)$ matrix.

3.6.4 Summary

In general, there were many issues with the RLS algorithm implementation on the multivariate DFE. Most literature like [43, 44, 39, 38] show end results, typically error and BER curves. However, the focus of this thesis is not only the performance investi-

gation of these algorithm but its implementation on the Tait platform. To ensure that implementation is possible, studying intermediate values like $\mathbf{P}(k)$ is essential. The later Chapters will illustrate some of these issues.

In summary, the observations and descriptions in this Section showed that $\mathbf{P}(k)$ can diverge as a result of certain matrix properties. From SISO analysis in the base simulations, there were correlations between the input to the feedforward and feedback filters. This correlation caused the divergence and was mitigated by adding decision-delays in the DFE. However, this method did not work for the Tait simulations and therefore, resetting the $\mathbf{P}(k)$ matrix is the only viable method for preventing the matrix from diverging without causing instability.

3.7 FIXED POINT ARITHMETIC

Fixed-point simulation is another major part of the performance investigation of these adaptive algorithms. The conversion of the floating-point simulation to fixed-point requires understanding of the FPGA that is used on the Tait platform, namely the Stratix EP1S125. The limitation on the FPGA is the precision, which is set at 16−bit for the LMS algorithm implementation.

During the implementation of these algorithms in fixed-point, numbers were scaled to keep them within a reasonable range to avoid undesirable operations like overflow and saturation. In [19], fixed point (or finite-precision) effects on the adaptive filter algorithms are discussed. The LMS algorithm is shown to be *numerically robust* compared to the RLS algorithm. However, various techniques were suggested to mitigate the effect of fixed-point arithmetic using variants of the RLS algorithm, for example QR-RLS and a symmetry-preserving RLS algorithm. However, it states that the square-root filters like QR-RLS are expensive to implement and very awkward to calculate. If there are issues with instability relating to fixed-point implementation, the easiest technique to apply is the symmetry-preserving RLS algorithm which preserves the symmetry of the $\mathbf{P}(k)$ matrix which is almost the same method as resetting the $\mathbf{P}(k)$ matrix. The literature explains that the instability is traced back to the Riccati equation because it is the computed difference between two nonnegative definite matrices as shown in the equation below:

$$\mathbf{P}(k+1) = \Big(\mathbf{P}(k) - \mathbf{K}_p(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{P}(k)\Big)\lambda^{-1}$$

During adaptation, $\mathbf{P}(k)$ loses the property of positive definiteness or Hermitian symmetry causing numerical divergence. Again, the $\mathbf{P}(k)$ matrix plays an important role in maintaining stability of the RLS algorithm in fixed-point. Therefore, the effects of fixed-point implementation of the RLS algorithm on the multivariate DFE will be observed. Results are illustrated in Chapter 6.

3.8 TRACKING

Tracking is the term used for adaptive algorithms when they reach a steady-state value. This can be easily observed from the error curve when the algorithm reaches the error floor. The error floor is basically the minimum value of error the algorithm achieves. In [19], tracking is explained as a *steady-state phenomenon* which is not the same as convergence. The rate of convergence and tracking capability are two different properties of the algorithm.

In summary, [19] mentions that the LMS algorithm exhibits a more robust tracking behaviour than the RLS algorithm. It even states that for time-varying communication channels, the LMS algorithm is preferable over the RLS algorithm due to its simplicity and tracking capability. Therefore, a hybrid between the two algorithms could be considered, where RLS algorithm is used for convergence (training mode) and LMS algorithm is used for tracking (decision-directed mode).

3.9 SUMMARY

Overall, this Chapter explained the various issues with the adaptive algorithms and the multivariate DFE. The focus was on RLS algorithm implementation issues which included stability and feasibility. The results of the simulations are shown in Chapters 5 and 6 which present two different simulation scenarios. Firstly, the base simulation where the LMS and RLS algorithm are investigated using basic specifications. The next simulation is similar in fashion except that Tait specifications are used. The simulation includes fixed-point simulation using the real-time channel data extracted from the Tait platform.

Generally, the RLS algorithm is a fast-converging algorithm. It is an alternative to the LMS algorithm and this thesis delves into its performance and feasibility. The concept of the multivariate DFE is described and how the adaptive algorithms are implemented using normal equations. Issues regarding the initial study of the RLS algorithm, the use of decision-delays and the correlation matrix $\mathbf{P}(k)$ in the RLS algorithm are also described. The implementation of the RLS algorithm is possible if certain mitigation techniques are applied if instability occurs.

Chapter 4

TAIT RESEARCH PLATFORM

4.1 INTRODUCTION

The purpose of the thesis is to investigate the performance of various adaptive algorithms for equalisation on the Tait MIMO research platform, often referred to hereafter as the CF3. CF3 refers to the version of the platform, representing *Configuration* 3. The platform has already been distributed and demonstrated in many educational institutes in New Zealand and around the world. Further information concerning this research platform can be found in [2]. The novelty of this platform lies in its ability to investigate the real-time aspects of wireless communications, hardware limitations and evaluate the effects of a real-time channel.

In addition, the CF3 allows data acquisition for offline signal processing without using the hardware. Generally, the system consists of a transmitter and a receiver which is currently set to 4 transmit and 4 receive antennas. The system is designed to be reconfigurable, therefore the CF3 can be expanded up to a 12 by 12 system and various algorithms for sychronisation, modulation scheme and adaptive filters can be implemented and tested. The system is heavily optimised for parallel processing pipeline, by taking advantage of the fast processing clock through pipelining to reuse multipliers and build on the concurrent nature of the FPGA.

4.2 SPECIFICATIONS

The CF3 system has the following specifications given in Table 4.1. The current setup of the Tait platform is 4 by 4 but it can be expanded to a 12 by 12 system as mentioned in [2]. The modulation scheme is $\pi/4$-DQPSK and is described in Section 4.4.1. The decision-device is an 8PSK quantiser and is explained in Section 4.4.3. The roll-off factor on the pulse-shaping filter is set at 100% which is suitable since the transmitted data are oversampled by 2. The reason for oversampling is based on the Nyquist theorem to avoid aliasing and it also reduces the sensitivity of the receiver to synchronisation errors thus improving detector performance [2]. The synchroniser is described in Section 4.4.2. The current adaptive algorithm being tested on the platform is the

System parameter	System setting
Number of transmit antennas	4
Number of receive antennas	4
Modulation scheme	$\pi/4$-DQPSK modulation
Pulse shaping	100% RRC filter
Decision device	8PSK Quantiser
Precision	16-bit complex
Sample rate	2 MHz
Symbol rate	1 MHz
System clock	120 MHz
Adaptive algorithm	LMS
Feedforward taps	4
Feedback taps	4

Table 4.1 Current specifications for the CF3

LMS algorithm. It has two different values of the step-size parameter μ. The multivariate DFE has two modes of operation, training and decision-directed mode. During training mode, $\mu = 0.0625$ and during the decision-directed mode, $\mu = 0.125$. These are the optimal parameters established after testing and are base 2 numbers for the purpose of hardware implementation. Selecting a smaller μ during training mode allows more stability to allow better filter weights adaptation. This is at the expense of slower convergence. The received data are complex numbers set to 16-bit precision which is reasonably accurate and it also allows a generous amount of multiplers to be used per clock cycle on the FPGA. The multipliers on the FPGA utilise DSP blocks on the FPGA described later in Section 4.3. These DSP blocks are features of the Stratix FPGA.

4.3 TRANSMITTER

The purpose of the transmitter is to produce packets suitable for transmission. As we know from the specifications, there are 4 streams of data, each stream corresponding to one antenna. Each packet consists of training sequences, synchronisation bits and payload data. Each of these streams of data have their own unique training sequences and synchronisation bits. The streams of data are then pulse-shaped and modulated ($\pi/4$-DQPSK) before transmitting them out of the antennas. In addition, the transmitter allows several transmit power settings, ranging from -3 to 15dBm. The transmitter also allows any of the 4 transmit antennas to be turned off for debugging purposes.

4.4 RECEIVER

Unlike the transmitter, the receiver is described in more detail because of the adaptive multivariate equaliser. There are three major parts to the receiver which are summarised in the following points.

1. Front-end RF Filtering
2. Packet Detection
3. Adaptive Multivariate Equalisation

The front-end RF filtering section includes an AGC module for amplitude adjustment and root-raised cosine matched filters. The packet detection is a synchroniser which is a unique part of the system, specifically designed by Tait Electronics. The data then goes through the decision feedback equaliser. The whole process is monitored through a controller. A block diagram below summarises the whole receiver design as shown in Figure 4.1. This particular block diagram is part of the FPGA section in the CF3. There are other modules that control the receiver but they are not relevant here. The first modules are the matched filters commonly found in software defined radios.

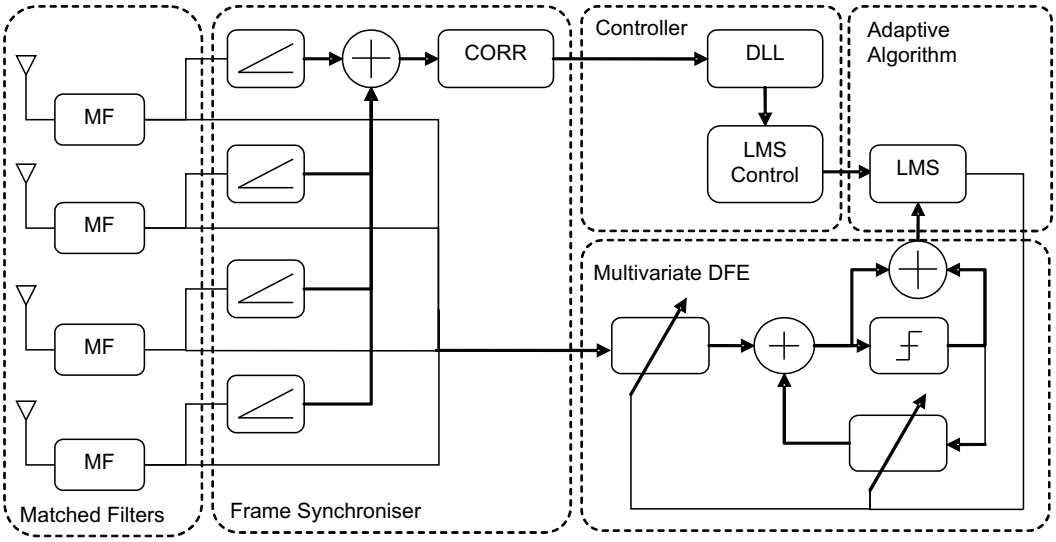

Figure 4.1 Block diagram of the receiver

The purpose of the matched filters is to maximise SNR at the sampling point of the bit stream. The next module is the frame sychronisation circuitry which is for detecting the packets that arrive at the antennas so that each sample can be indexed. The information regarding the location of the start of the packet is fed into the controller module which administers the process for adaptive equalisation. The final modules are used for adaptive equalisation, which consists of the adaptive algorithm and multivariate DFE

module. The adaptive equalisation consists of two process, adaptation and filtering. The adaptation process updates the filter weights on the DFE every symbol. The filtering process consists of multiplying the inputs with the filter weights.

4.4.1 Modulation

The modulation used in the CF3 system is $\pi/4$-DQPSK. The constellation for $\pi/4$-DQPSK is illustrated in Figure 4.2. It is an 8-point constellation that is essentially made up of 2, 4-point (or QPSK) constellations that are offset by $\pi/4$. Symbol phase is alternately selected from one of the two QPSK constellations and as a result, successive symbols have a relative phase difference that is one of the four angles, $\pm\pi/4$ and $\pm 3\pi/4$. Switching between the two constellation sets every succesive symbol ensures that there is at least a phase shift which is an integer multiple of $\pi/4$ radians between successive symbols. This ensures that there is a phase transition for every symbol which reduces the complexity of symbol timing recovery and synchronisation. In general, $\pi/4$-DQPSK

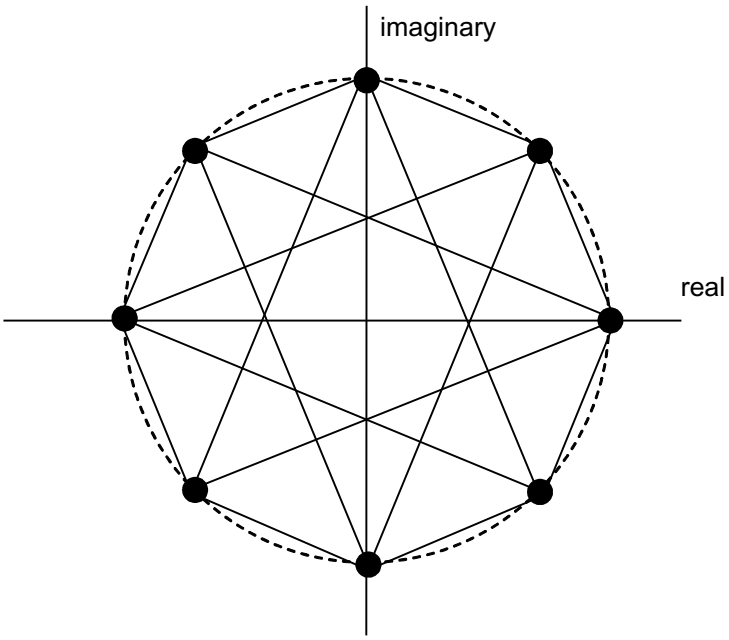

Figure 4.2 $\pi/4$-DQPSK constellation points showing the phase transitions

is a reasonable and practical modulation scheme due to its immunity to frequency drift and its non-coherent detection method. In addition, the modulation scheme provides a novel approach to the frame synchronisation circuitry, which provides benefits during packet detection as shown next in Section 4.4.2

4.4.2 Synchronisation

The synchroniser is an integral part of the receiver for packet detection. In the packet, a specified number of bits called synchronisation bits are set for packet detection. The synchronisation method uses an angle detector and MRC-like summation prior to correlation and symbol phase locking. In general, it measures the energy of the packet which corresponds to a certain index in the transmitted sequences. The controller uses this index to calculate the locations of the start of the packet. The index of the synchronisation bits indicate where the synchronisations bits are located to inform the controller where the start of the packet is. Therefore, for the data entering the DFE, the controllers can determine where the start of the packet is, via the indexing. As mentioned previously, the modulation scheme used for the Tait platform is $\pi/4-$DQPSK. The modulator is implemented using the equations below.

$$s(k) \in \{e^{(\pi/4)}, e^{j(3\pi/4)}, e^{(-\pi/4)}, e^{j(-3\pi/4)}\} \tag{4.1}$$

$$u(k) = u(k-1)s(k) \tag{4.2}$$

To produce a $\pi/4-$DPQSK constellation, we form a normal QPSK signal as shown in (4.1). Then apply the differential encoding as shown in (4.2). The differential decoder can be derived from (4.2) to be a complex division and by assuming that the receive signal has unity gain. Therefore, the differential decoder can be implemented as a delay line feeding a complex conjugate and multiply circuit.

$$s(k) = \frac{u(k)}{u(k-1)} = \frac{u(k)u^*(k-1)}{|u(k-1)^2|} \tag{4.3}$$

This decoding approach would not work with a MIMO signal as the multiplication is not linear and the resulting signal is no longer a linear superposition of multiple QPSK signals. However, this particular decoder is useful for detecting the packet, i.e frame synchronisation. This is during the time when the receiver cannot perform any equalisation or phase correction on the signal due to the lack of training. For the training and payload data, the magnitude of the output from the decoder is small. However, the output magnitude for the synchronisation bits is larger, therefore providing a novel packet detector.

4.4.3 Decision Device

One of the most interesting parts of the multivariate DFE in the Tait platform is the decision device. There are two decision devices, one used for the actual output for displaying the QPSK constellation and the other for the input to the feedback filter. The QPSK output utilises a differential decoder and a QPSK slicer. The inputs to the feedback filter utilise an 8PSK quantiser representing the 8 points on the $\pi/4-$DQPSK

constellation.

Unlike the feedforward filter which receives complex numbers of varying magnitude and phase, the feedback filter receives only 8 pre-defined states. This in turn allows lower resolution in the multiplier and less hardware resource consumption. In general, the 8PSK quantiser takes a soft complex input and maps it into a 3-bit output which represents the closest point on the 8PSK constellation. A look-up table is used to store the 8PSK values, which is equivalent to storing them on RAM blocks on the FPGA. These RAM blocks are described later in Section 4.5.5.

From previous research in [2], the 8PSK quantiser is a superior decision device for the feedback filter inputs for low signal conditions. These low signal conditions are caused by non-linearities and the feedback loops. Using a $\pi/4-$DQPSK modulated signal as the input may not prove to be stable and can cause error propagation.

4.4.4 Reconfigurability

One advantage of the CF3 is its reconfigurability. The current 4 by 4 system can be expanded up to a 12 by 12 system by adding two more 4-channel digital signal processing platforms on the transmitter and receiver. The packet structure illustrated in Figure 4.3 can also be modified in terms of length and content. The packet has pre-defined training sequences which can be orthogonal as proposed in [44, 43] for suppressing interference and allowing better convergence in a frequency selective channel.

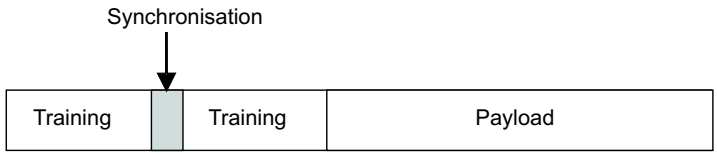

Figure 4.3 Example of the packet structure used in CF3

4.4.5 Automatic Gain Control

Automatic Gain Control (AGC) is required in communication systems to help maintain the received signal strength at a reasonable range. The signals fluctuate at the antenna because of movement, positioning and location of the platforms. The AGC module is a front-end block at the receiver on the Tait MIMO platform. In the CF3, there are 4 LEDs representing each received data stream which indicate if amplitude adjustment has been applied to it. The LEDs help indicate the suitability of the position of the receiver. Without the AGC module, the inputs to the DFEs are unusable due to the extreme variations in their values.

4.5 FPGA DEVICES

4.5.1 Introduction

The FPGA belongs to the family of programmable logic devices. The main reason for using FPGAs lies in their parallel processing capability, enabling them to perform more operations than DSPs. FPGAs allow designers to implement various designs in hardware for testing and prototyping, in this case different hardware blocks or modules on the CF3 system. In general, the FPGA is made up of configurable logic blocks (CLB)

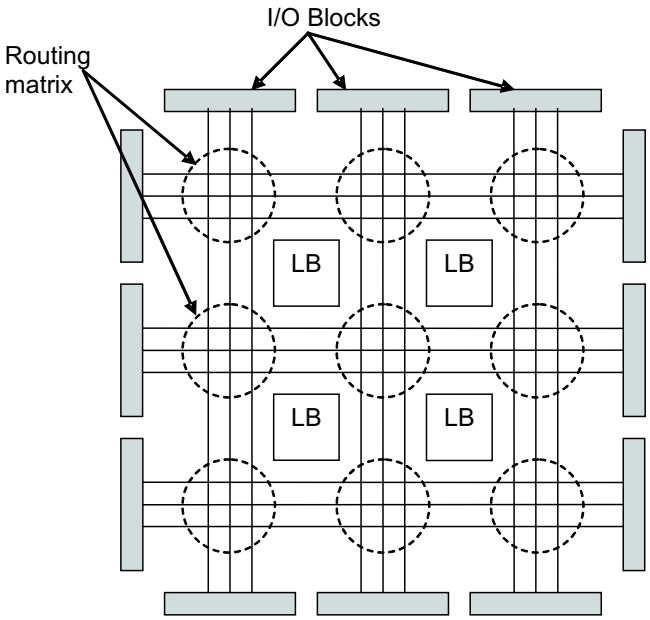

Figure 4.4 A typical FPGA architecture

arranged in an array with interspersed switches that can rearrange the interconnections between logic blocks. A typical FPGA architecture is shown in Figure 4.4. Surrounding the FPGA are input/output (I/O) pads. Each CLB consists of a 4-input look-up table (LUT) and a flip flop. Each CLB can be individually programmed to perform a logic function. The switches are then programmed to connect these blocks together to form a complete logic function.

4.5.2 Altera Stratix

The FPGAs currently used to implement the adaptive multivariate equaliser on the CF3 belongs to Altera's range of high density FPGAs, Stratix. Stratix FPGAs have special features like DSP and RAM blocks which are easy to implement on the FPGA and do not consume any logic elements (LE). Further details regarding these special

features are described later in this Section. The particular model being used is the Stratix EP1S25, which is a medium-level FPGA used for most of the signal processing work on the Tait platform.

With communication applications like adaptive equalisation, signal processing operations are prevalent. The storage of filter weights and data buffering can be easily implemented using RAM blocks. Complex applications like multiplication and accumulate can be implemented using DSP blocks which save resources. Mathematically intensive operations like adaptive equalisation are therefore simple to implement on the FPGA using these special features on the Stratix. In addition, the software used for resource and timing analysis on the FPGA is Quartus II.

The Stratix EP1S25 has the following specifications as shown in Table 4.2. Other important details such as I/O pins, package and supply voltage are available in [82]. The main concern is the amount of resources available for RLS algorithm implementation. There are various optimisation methods currently used in the FPGA, for example reusing multipliers and blocks to reduce the usage of logic elements. Basically the motto of *less is more* is used. The memory available in the Stratix EP1S25 is sufficient to store all the filter weights and matrix elements. The DSP blocks might not be sufficient to implement the RLS algorithm which is discussed later in Chapter 7. If the RLS algorithm is unable to fit into the EP1S25, the implementation and testing is still possible through simulations.

Logic Elements	25560
RAM bits	1944576
DSP blocks	10

Table 4.2 Summary of the EP1S25

4.5.3 Software Language

In the FPGA, the multivariate DFE is implemented using code written in VHDL, and it is tested using software readily available from Stratix called Quartus II. VHDL is a hardware level language like assembly language. It is a verbose and complex language, and requires a good understanding of digital components. One of the major features of VHDL is its ability to allow electrical aspects of circuit behaviour to be described, for example rise and fall times of signals and delays through gates. Another major feature of VHDL is its ability to be used as building blocks for larger and more complicated systems, therefore a hierarchal system can be designed. This hierarchal system is applied in the design and implementation of the adaptive multivariate DFE on the Tait platform. The VHDL designs are simulated using the ModelSim software.

4.5.4 DSP blocks

DSP blocks are dedicated blocks in the Stratix FPGAs. DSP blocks allow faster performance and reduce development time since distributed arithmetic is not required to produce the multipliers, therefore no resources are consumed. Evidently, these blocks are designed for signal processing purposes, for example FIR (Finite Impulse Response) and FFT (Fast Fourier Transforms). Each DSP blocks represent a certain number of real multipliers at various precision as shown below in Table 4.3.

Number	Precision
80	9-bit multipliers
40	18-bit multipliers
10	36-bit multipliers

Table 4.3 The maximum number of real multipliers based on precision

4.5.5 RAM blocks

RAM blocks are also dedicated blocks in the Stratix FPGAs. They are used mainly for storing data, in this case filter weights, global variables, matrix entries etc. The size of the memory on the FPGA is sufficient for the adaptive multivariate DFE. RAM blocks can also be used for buffering and storing data for offline processing. Typically, another FPGA would be required for the sole purpose of storing data. The RAM blocks are divided into three categories as shown in Table 4.4, each representing the capacity of the block.

Number	Type	Size
3798	M512	512 bits
422	M4K	4086 bits
3	M-RAM	512 kbits

Table 4.4 RAM blocks on the Stratix EP1S25

4.6 SIMULATION, CHANNEL MODELS AND MEASUREMENTS

4.6.1 Simulation Scenarios

Table 4.5 gives a summary of the simulation stages for adaptive algorithm testing and implementation in this thesis. Details regarding the $\pi/4$-DQPSK modulation are given in Section 4.4.1 and synchronisation is discussed in Section 4.4.2. The channel models are described in Section 4.6.2.

Name	Location	Channel Model	Modulation	Miscellaneous
Base	Chapter 5	Jakes Model	BPSK	None
Tait (floating point)	Chapter 6	Quasi-stationary	$\pi/4$-DQPSK	Synchronisation
Tait (fixed point)	Chapter 6	Quasi-stationary	$\pi/4$-DQPSK	Synchronisation
Tait (fixed point)	Chapter 6	Real-time (Captured data)	$\pi/4$-DQPSK	Synchronisation

Table 4.5 A summary of the different simulation scenarios

4.6.2 Channel Models

Overview

This Section will briefly go through the channel models used in the simulations. As seen previously in Table 4.5, the simulation scenarios assume 3 different channel models.

Jakes Model

The Jakes model [83] is a widely accepted model in simulating the urban wireless channel. The Jakes method is often used due to its simplicity of implementation and its statistical reliability [1]. The Jakes model employs a sum of sinusoids method which is dependent on the Doppler frequency. This frequency determines the speed of the simulated device.

Consider the channel coefficient $g(t)$ of a SISO link at a particular time t. To devise the Jakes model for simulation of wireless communication channels, $g(t)$ is separated into two components, the real (or in-phase) and imaginary (or quadrature) components. Each component is a zero-mean independent Gaussian process. Equations (4.5) and (4.6) show the sum of sinusoids approach used by the Jakes method.

$$g(t) = g_I(t) + jg_Q(t) \tag{4.4}$$

$$g_I(t) = \frac{1}{\sqrt{N_0}}\left(\sum_{n=0}^{N_0-1}\cos(\omega_M \cos\alpha_{nk}t + \phi_n)\right) \tag{4.5}$$

$$g_Q(t) = \frac{1}{\sqrt{N_0}}\left(\sum_{n=0}^{N_0-1}\sin(\omega_M \sin\alpha_{nk}t + \phi_n)\right) \tag{4.6}$$

where $k = 0, 1, 2, \ldots, M-1$ and $n = 0, 1, 2, \ldots, N-1$. M represents the number of independent fading waveforms that are required, each of which is generated by N incident waves with random phases ϕ_n and equally spaced arrival angles around the

moving receiver. Other important equations follow:

$$N_0 = \frac{N}{4} \tag{4.7}$$

$$\alpha_{nk} = \frac{2\pi n}{N} + \frac{2\pi k}{MN} + \frac{\pi}{2MN} \tag{4.8}$$

$$\omega_M = 2\pi(fv/c) \tag{4.9}$$

where f is the carrier frequency, v is the vehicle speed and c is the speed of light. N_0 is the waveform index. A more in-depth description regarding the Jakes model is shown in [84, 85, 70, 86]. This literature contains variations or different approaches to the original model seen in [1, 83]. However, the models still hold onto the basic principle of generating multiple Rayleigh fading waveforms by superimposing multiple sinusodial waveforms.

Quasi-Stationary Model

Another model used in the simulation is the quasi-stationary model. This model changes in terms of gain and phase for each block of samples to simulate a time-varying channel. This model is termed quasi-stationary because it is a static channel that changes for every block of samples. The variations in phase and magnitude are not the same as for the Jakes model but are easier to understand and implement. The initial testing of the Tait platforms were done in stationary environments, because the platforms (both transmit and receive) are placed in a fixed position inside the Tait research and development building. Therefore, changes in amplitudes are not implemented in the model. However, random phases were included to create slight variations.

Real-Time

Measurements of a wide variety of channels are difficult to obtain. It can be costly and time-consuming to do measurements of the channel. Therefore, suitable channel models were used to simulate the performance of the adaptive multivariate DFE. These models were described previously in Section 4.6.2 and 4.6.2. Nevertheless, some real measured data is used for the received signals in Chapter 6 and this obviously includes the effects of the real channel.

4.6.3 System Simulation and Debugging Outputs

For the CF3, there are several outputs for debugging and analysis. Shown in Figure 4.5 is the simulation and verification procedure. The process chart shows how the verification of the system is done. Simulating the algorithms in Octave/Matlab allows a multitude of parameter and variable modifications to test the multivariate DFE. In the

process, offline processing can be done by extracting data from the FPGA and storing the data as files on the server. These files contain various parameters of the DFE including input vectors, error vectors, outputs from the DFE etc. Input vectors to the DFE are received data that has passed through the front end RF block. Error vectors are differences between the input and output of the decision device, except during training mode when it is the difference between the input to the decision device and training data. Evidently, these text files come in 4 streams representing the 4 transmitted packets and by running shell scripts, the data from the FPGA can be converted to files that are recognisable in Matlab to allow it to be processsed and analysed. The next three Figures show the major outputs produced from the FPGA for analysis on Matlab. The Figures are divided into four parts, each part corresponding to the results produced from each DFE arranged as $\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}$. Figure 4.6 shows the decoded soft output for the 4 streams resembling a QPSK constellation. Figure 4.7 shows the soft equalised output for the 4 streams resembling the 8 points from a $\pi/4$-DQPSK modulation. These outputs are inputs to the decision device. Figure 4.8 shows the error curves for the 4 streams. The Figure shows that the error curves are converging, all at different rates, and demonstrates the behaviour at steady-state. These outputs are produced using data from the real-time channel. The location of these platforms were in the Tait Development building, therefore the channel environment is relatively static. By closely observing the error curve starting at 256^{th} symbol, there is a temporary reduction in its magnitude due to the transmission of synchronisation bits. The synchronisation bits are transmitted in-phase on all transmit antennas, therefore producing a much stronger and cleaner signal.

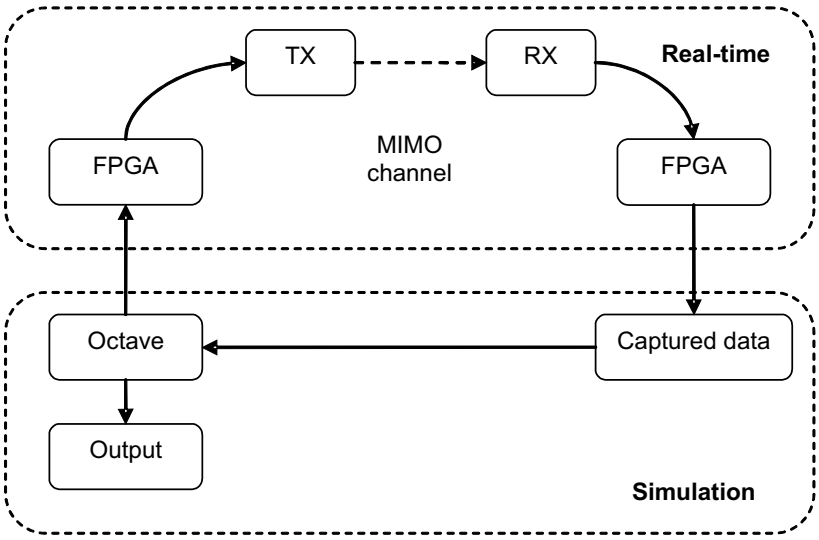

Figure 4.5 System simulation and verication process chart

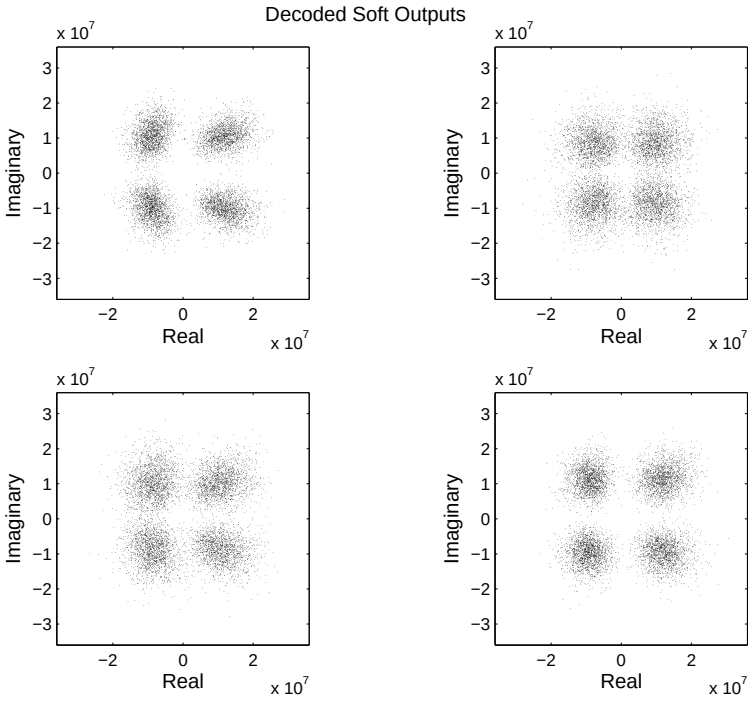

Figure 4.6 Examples of decoded soft output from each antenna

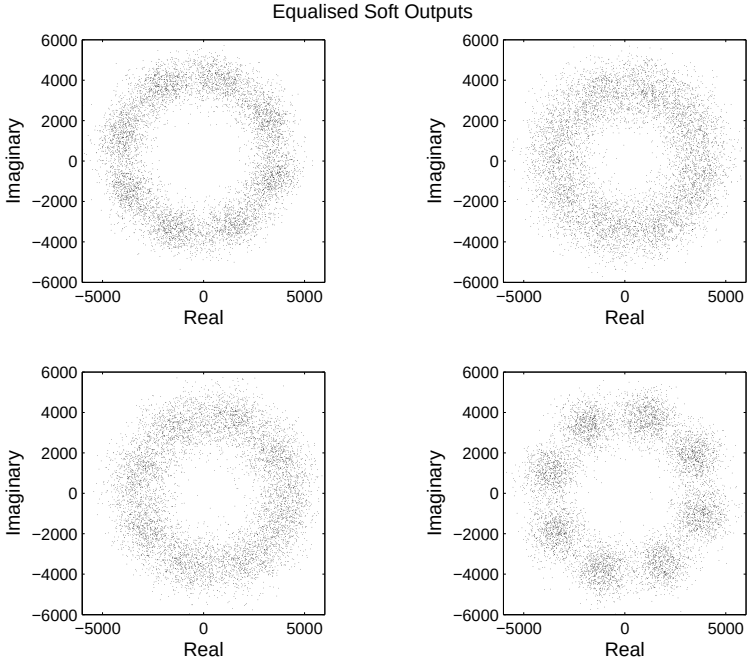

Figure 4.7 Examples of the equalised soft output from the DFE

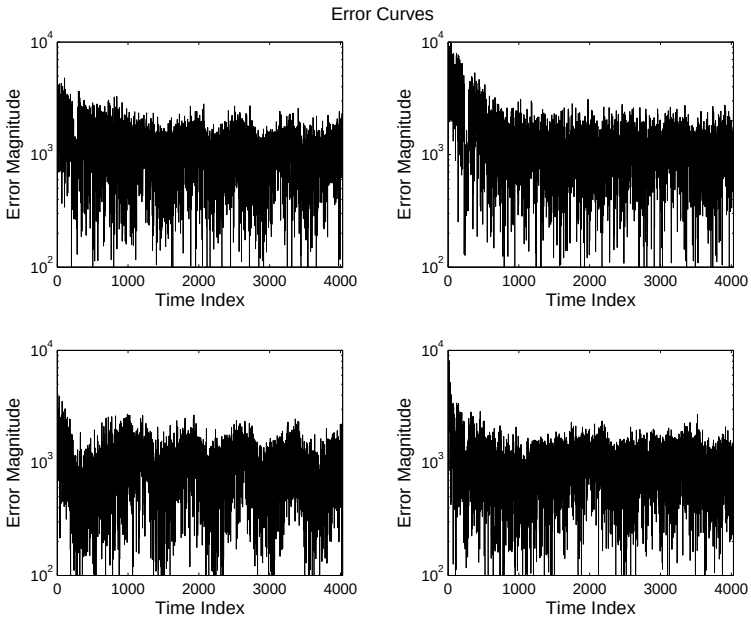

Figure 4.8 Examples of the error curve from each antenna

4.7 HARDWARE ASPECTS

4.7.1 Introduction

Certain hardware aspects have to be taken into account regarding the implementation of these algorithms. The bit-precision of variables like filter weights and input data are currently set to 16 bits. Increasing the precision requires more resource consumption from the FPGA. As we know, the FPGA has DSP and RAM blocks that are limited to a certain number. For example, the Stratix EP1S25 has 10 DSP blocks, which represent about 40 16-bit multipliers as seen in Table 4.3. These are the maximum number of multipliers per clock cycle. Note that these multipliers are not only used for DFE purposes but for other applications on the platform, for example, the synchroniser.

4.7.2 Precision

The precision for the multivariate DFE is 16-bit, allowing the use of the 18-bit multipliers in the Stratix FPGA. The feedforward filters are implemented using 16-bit two's complement arithmetic therefore producing a 32-bit multiplier output. As mentioned before in Section 4.4.3, the feedback filter encodes the decisions into 3-bit numbers. It generates a 36-bit output for subtraction before the decision device but internally, performs the arithmetic at only 21-bit resolution.

Scaling of these fixed point numbers are required before each multiplication. Therefore, a 32-bit number has to be scaled back to its equivalent 16-bit representation to allow

that number to be multiplied. The scaling is done by taking only the most significant bits and discarding the least significant bits. This is the drawback of implementing any integer in hardware. It is important to make sure the range of these integers do not drift to avoid overflow, which can cause the number to saturate. A number that goes through saturation will most likely produce undesirable results.

4.7.3 Complex Multipliers

Complex multipliers are different from normal multipliers because complex numbers are made up of two parts, real and imaginary. The total number of real multipliers per complex multiplication is 4 which is also shown below in Equation 4.10

$$(A + jB) \times (C + jD) = AC - BD + j(AD + BC) \tag{4.10}$$

One complex multiplication requires 4 real multipliers. For the current implementation of 16-bit precision, there is a maximum of 40 18-bit multipliers per clock cycle. For a system clock of 120 MHz, this is approximately 4800 real multipliers per system clock cycle. The amount of multipliers is sufficient for the LMS algorithm implementation on the adaptive multivariate DFE.

Now consider the current specifications shown in Section 4.2. By taking into account the filter taps and the number of transmit and receive antennas, the multivariate DFE requires a total number of 128 complex multipliers which is equivalent to 512 real multipliers. For the LMS algorithm, each element in the input vector is independent when calculating the *change* during adaptation, i.e. error vector. Therefore, the processing can be split to allow a filtering function and a weight update function. The next Section illustrates this concept.

4.7.4 Multivariate DFE

The theory shown in Chapter 3 revealed that the multivariate DFE can be designed using multiple DFE blocks since their inputs are all identical. For the Tait platform, the LMS algorithm has been assimilated in the DFE block to form the LMS-DFE module. Figure 4.9 shows one of the LMS-DFE modules implemented on the Tait platform. The modules are used to construct the 4×4 DFE receiver through a structural VHDL design simply by generating the exact same modules 4 times [2]. Each module calculates the weights that corresponds to each transmit antenna, therefore each module has different training sequences in its RAM blocks. Figure 4.10 illustrates this circuit which contains one complex multiplier and two adders to be shared by all the filter taps as well as the LMS update algorithms. These functions are selected via one of the select lines shown. Based on the specifications shown in Section 4.2, there are two samples per symbol. This means that within each symbol period, there are two sample periods.

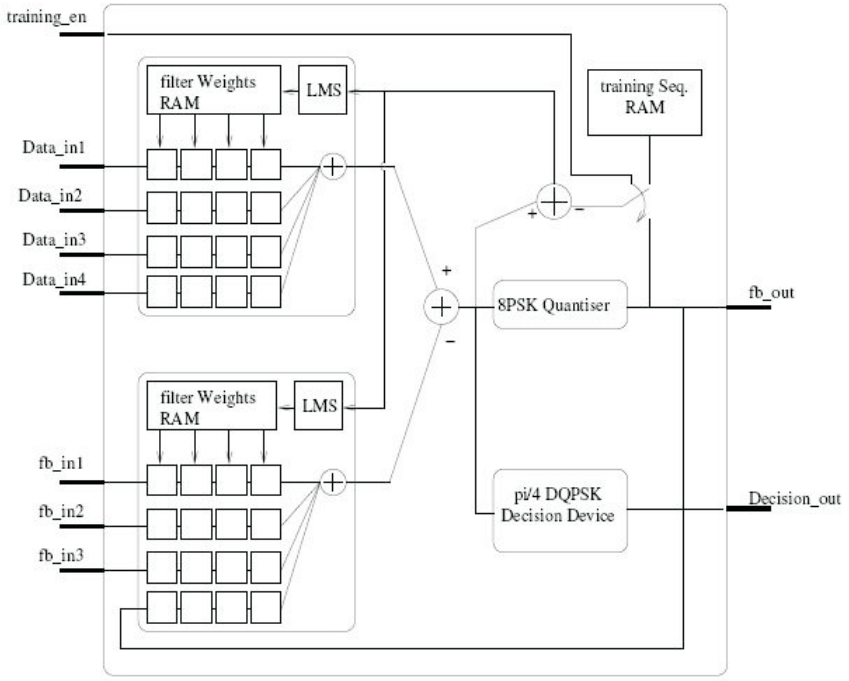

Figure 4.9 Internal structure of the LMS-DFE module for one DFE [2]

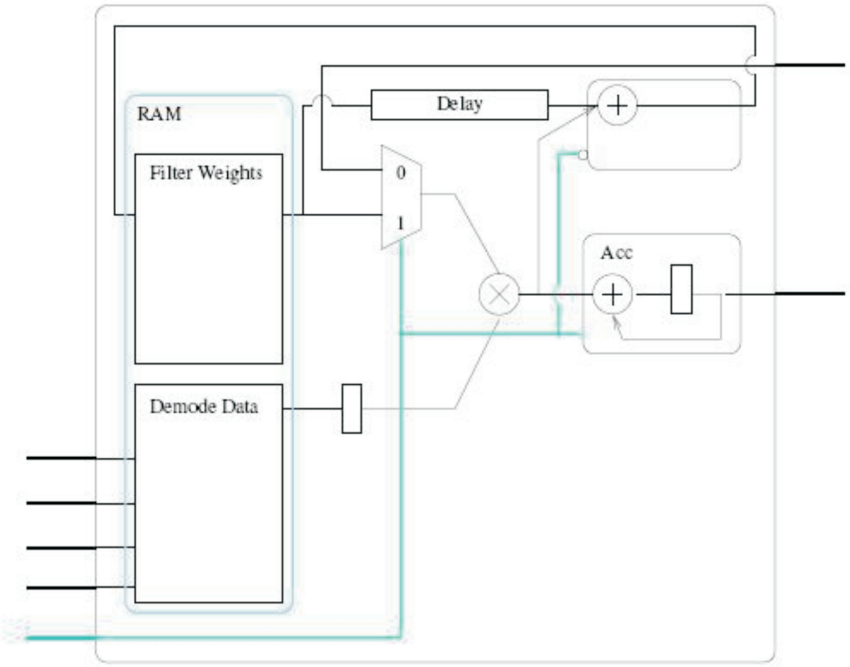

Figure 4.10 Internal structure of the DFE filters [2]

One sample period is dedicated to the filtering function and the other for the weight update function. For every odd sample clock, the system performs the multiply and accumulates for the filtering and for the even sample clock, the same circuit is used to multiply the data with μ and accumulates directly into the RAM block where the weights are stored.

4.8 SUMMARY

Extensive work was performed in the first stages of the thesis performing an initial study of the platform. Familiarisation of the real time platform required a general literature review of the documentation for the Tait platform covering its simulation and hardware systems, which included studying the various hardware modules. Instructions for operating the platform and obtaining measurements were also reviewed. Most of the technical information regarding the platform is not disclosed in detail to protect the intellectual property of Tait Electronics, therefore an overview was given in this Chapter.

Generally, the Tait research platform is a novel hardware system that allows various modifications in its design because of its reconfigurability. The system can be expanded up to a 12×12 system. The platform also allows investigation of real-time issues relating to the FPGA that are usually not considered during simulation and analysis of adaptive algorithm implementation for the multivariate DFE. The study of this MIMO system can be observed through the simulations and hardware platform that was implemented which are shown in the next few Chapters. The simulation provides flexibility and the possibility of analysis, and the hardware platform provides more insight into real-time issues. In general, the most important aspects of the Tait platform are described in this Chapter. Specifications and background theory for the Tait platform were given. Various design issues and techniques were described and discussed. This Chapter presents the background for the current system and leads to the next step of RLS algorithm implementation and analysis.

Chapter 5

BASE ADAPTIVE MULTIVARIATE DFE SIMULATION

5.1 INTRODUCTION

This Chapter describes the base simulation of the adaptive multivariate DFE, investigating specifically the behaviour of the adaptive algorithms, the structure of the multivariate DFE, the effects of the channel etc. In addition, the simulation tests various issues that might occur in the CF3 simulation in a simpler more controlled environment where properties of the algorithms can be studied in detail without being obscured by the effects of other processes such synchronisation and $\pi/4$-DQPSK modulation.

The simulation assumes BPSK modulation and the Jakes channel model [83]. A BPSK modulation is used to remove the unneccessary complexity that comes with $\pi/4$-DQPSK modulation when testing the multivariate DFE strucuture. In addition to that, a Jakes model is used to simulate any mobile characteristics that can affect the DFE, since the model is relatively easy to implement. Like the Tait simulations, the base simulation is flexible allowing various parameters to be modified. These parameters include the number of antennas, tap filter lengths, mode of operation etc.

In this Chapter, two adaptive algorithms, the LMS and RLS, are investigated. The various outputs include the error curve, BER, MSE and some important intermediate variables (required for hardware implementation). The error convergence curves help identify how quickly the algorithm can convergence to the *optimal* weights. The BER and MSE indicate how reliable the algorithms are. The intermediate variables indicate if there is any undesirable behaviour that might affect hardware implementation.

5.2 SIMULATION RESULTS

5.2.1 Introduction

Shown in this Section are the observed results from Matlab simulations of the adaptive multivariate DFE. The individual Sections will show the observed plots resulting from simulation when modifying various parameters. The first Section shows the simulation using 1 transmit and 1 receive antenna to observe the performance of the adaptive

SISO DFE. The following Sections describe the effect of various modifications to the multivariate DFE parameters such as SNR (Signal to Noise Ratio), tap lengths, Doppler frequency, decision delay and a discussion on using a hybrid mode. Of particular interest is any instability that might occur.

5.2.2 Brief SISO Analysis

The SISO DFE is the basic form of the multivariate DFE obtained by reducing the number of transmit and receive antennas of the multivariate DFE simulation to 1. The complexity of the DFE is drastically reduced when this happens and allows a clear investigation of the algorithms and debugging. The results shown here are for a simulation run with a packet length of 2000 and a training length of 500. The parameters are set as $\lambda = 0.99$ for the RLS algorithm, $\mu = 0.01$ for the LMS algorithm. Tap lengths were set at 4 for both the feedforward and feedback filters. Figure 5.1 shows one of the plots used to compare the performance of the two algorithms. The plot shows the size of the error. This is calculated as $x(k) - \hat{x}(k)$ and the mathematical notation is described and illustrated in Section 3.3.2. Figure 5.2 shows the rate of convergence of the largest tap weight for each algorithm. Figure 5.3 shows that the variation in $Re(H)$, the real part of the channel coefficient against time is almost static. The channel coefficient is produced from the Jakes model with a Doppler frequency f_D of 0.00005 rads^{-1}. Figure 5.4 is a typical error curve that defines the algorithm's performance. The results illustrated in this Section for the SISO DFE shows that the RLS algorithm outperforms the LMS algorithm by producing fewer errors and quickly converging towards the desired tap weight and error floor. These results are concurrent with the literature [19].

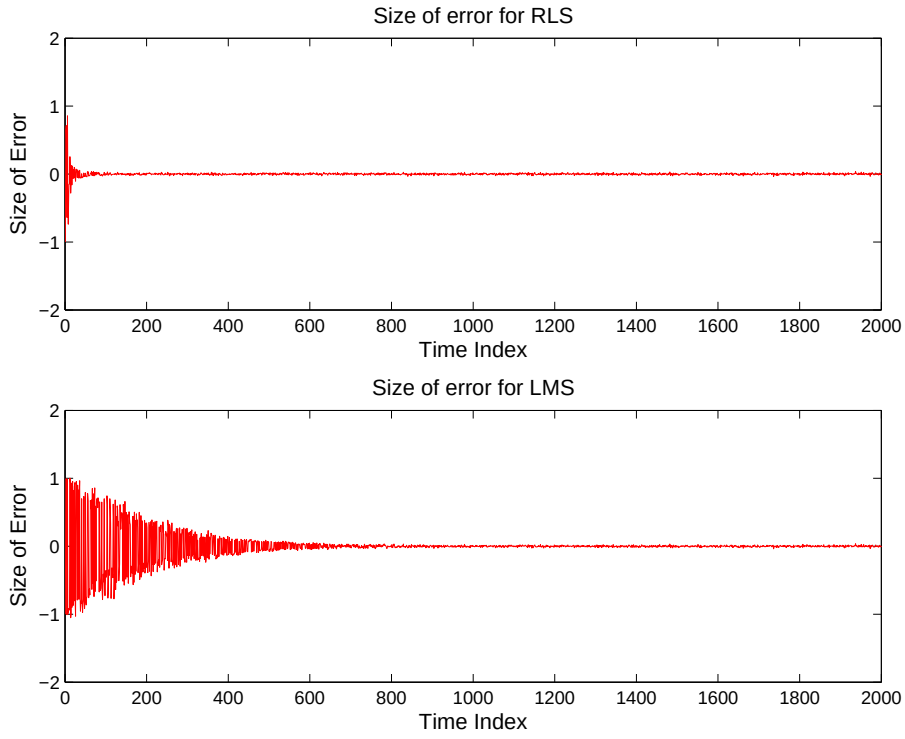

Figure 5.1 Comparison of the size of errors between the SISO-LMS and SISO-RLS algorithms

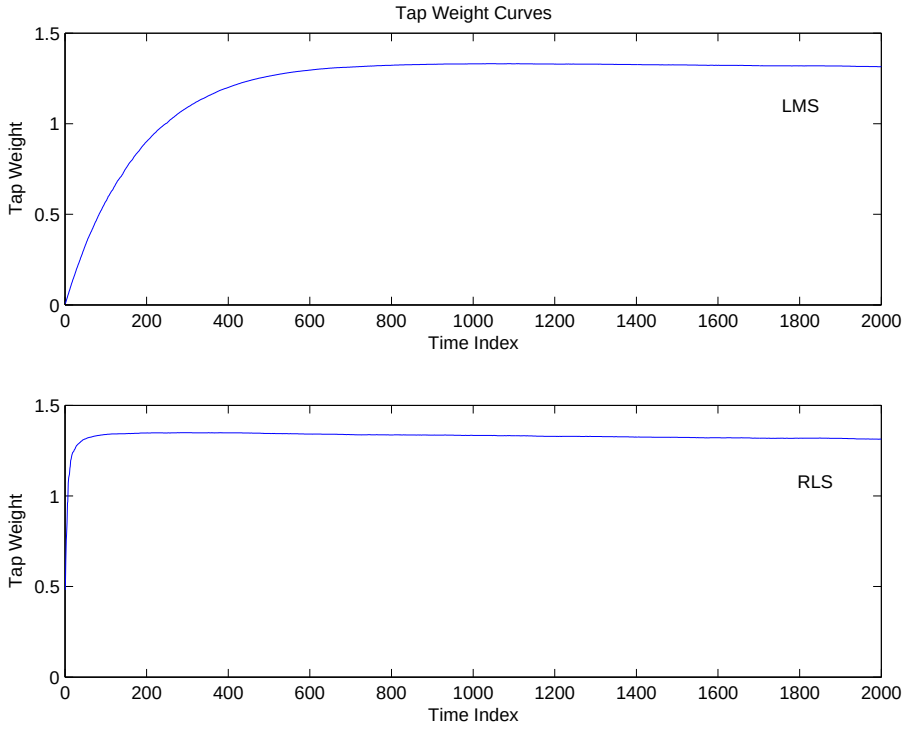

Figure 5.2 Comparison of the convergence rate of the weights for the SISO-LMS and SISO-RLS algorithms

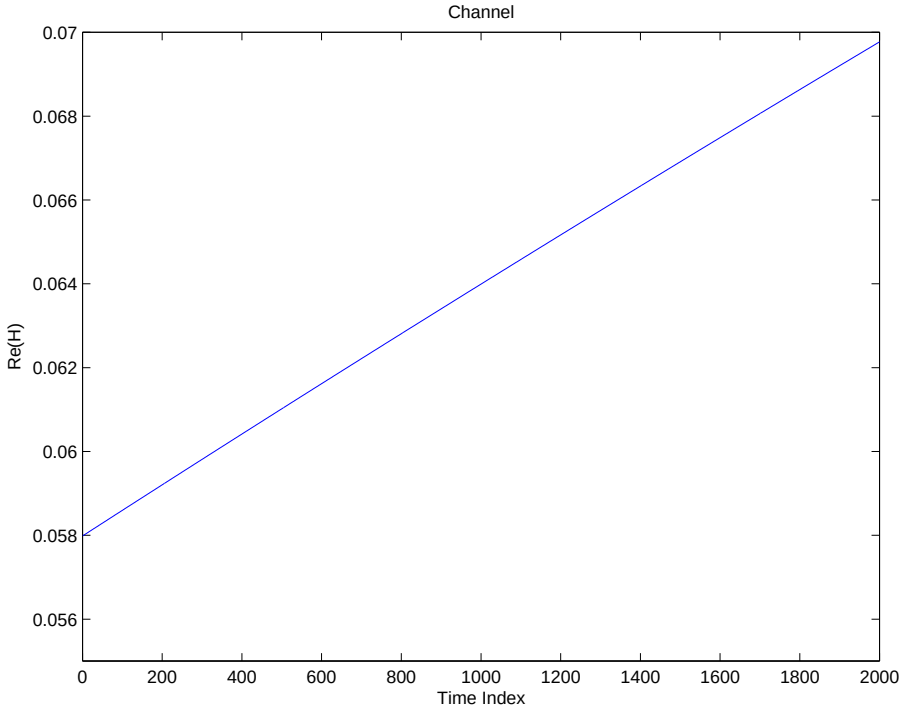

Figure 5.3 Real part of the SISO channel coefficient against time

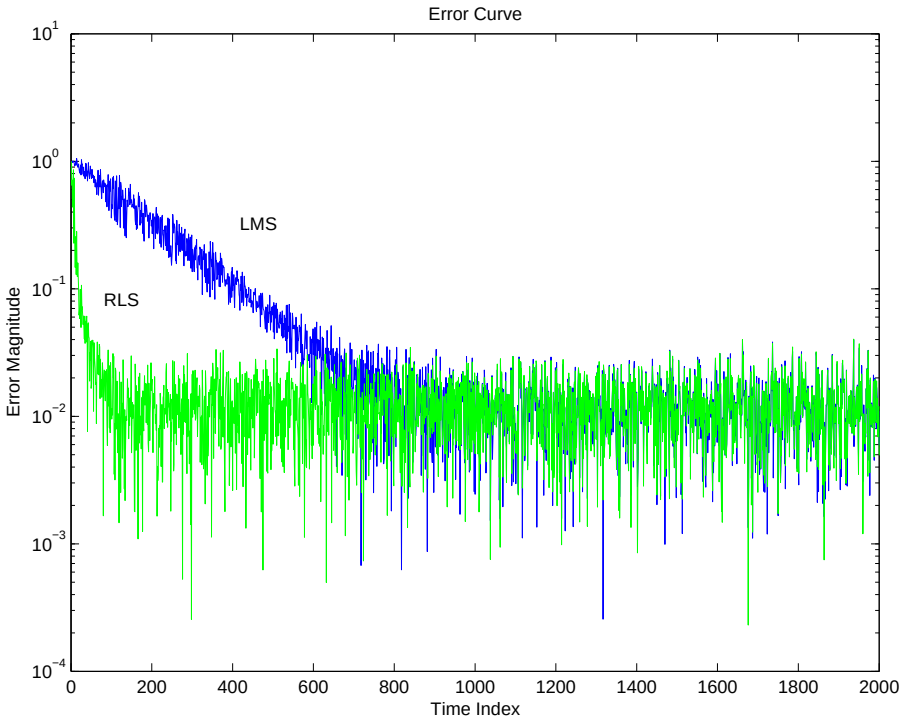

Figure 5.4 Comparison of the error curves of the SISO-LMS and SISO-RLS algorithms

5.2.3 MIMO Analysis

The previous Section described the adaptive SISO DFE simulations briefly, basically illustrating that the system works for a SISO environment. All work from this point onwards focusses on the adaptive multivariate DFE, where the number of transmit and receive antennas are 4. This Section illustrates the performance of the LMS and RLS algorithms in general by varying its SNR. Simulation parameters such as λ, μ, f_D and tap lengths are the same as in Section 5.2.2. The packet length is set to 8000 with a training length of 1000.

From Figures 5.5 and 5.6, the convergence rates are observed to be virtually the same for any SNR. The Figures show results from only 1 channel out of the 4 to closely observe the LMS and RLS algorithm behaviour. The major effect of the SNR is the error floor when the error curve reaches steady-state. Evidently from the Figures, the RLS algorithm outperforms the LMS algorithm for all three different SNR values; 20dB, 30dB and 40dB. The Figures use a moving-average to smooth the error curve and make the convergence rates and error floors more distinct. Figure 5.7 shows the average BER against SNR curve. Based on the diagram, it can be concluded that the RLS algorithm achieves much lower BER values than the LMS algorithm. The average BER means that the BER values for all 4 streams are averaged to gain an overall BER metric.

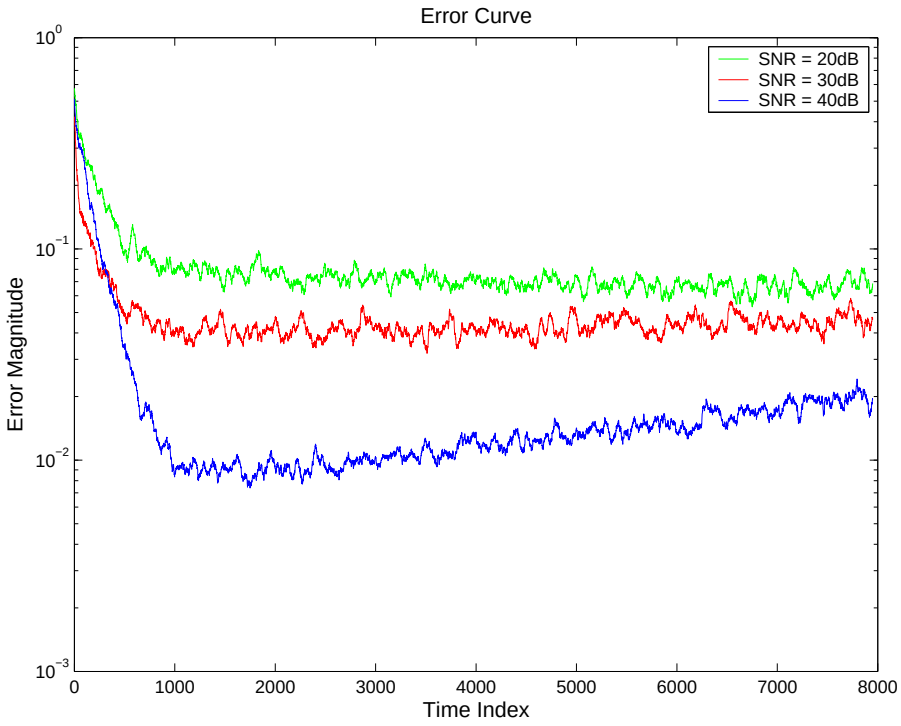

Figure 5.5 Moving-average of the error curves showing the different signal levels for the LMS algorithm for one channel

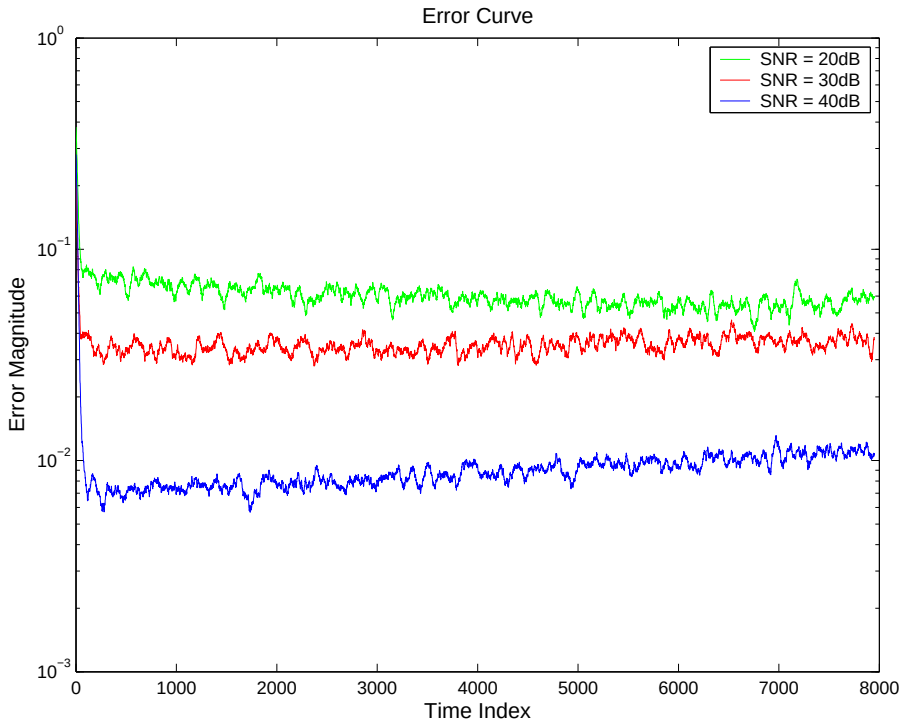

Figure 5.6 Moving-average of the error curves showing the different signal levels for the RLS algorithm for one channel

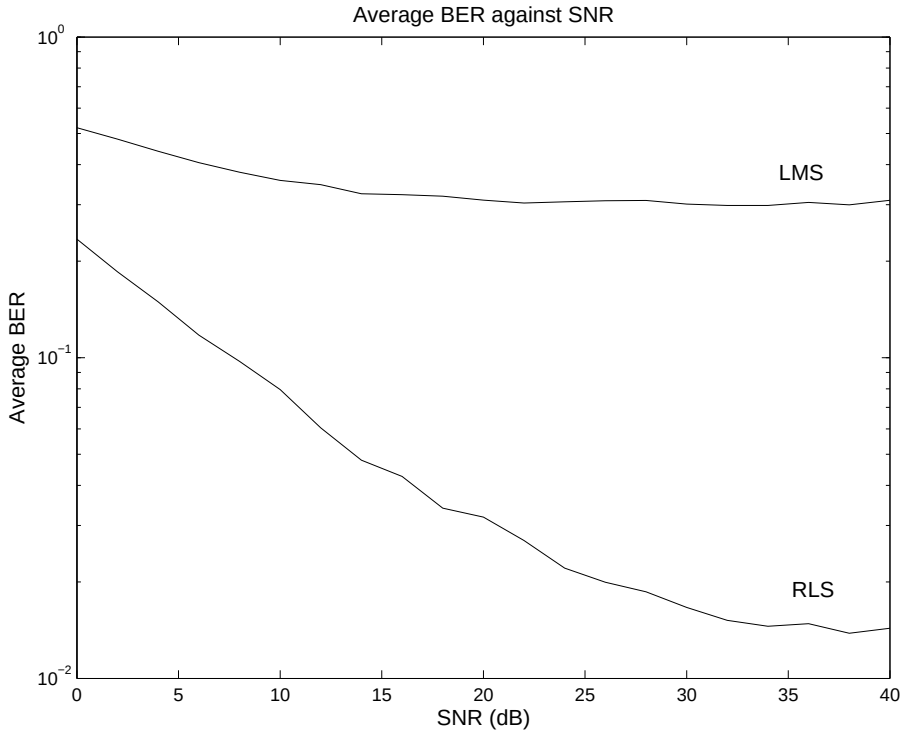

Figure 5.7 Average BER against SNR for the LMS and RLS algorithm

5.2.4 Filter Order

Filter order is expected to affect BER based on the results in [50] for a SISO DFE. For a multivariate DFE this may also have an effect on BER. Figure 5.8 shows the plot of averaged BER for all channels versus tap lengths. The algorithms were simulated with adaptive variables of $\lambda = 0.99$ for the RLS algorithm and $\mu = 0.01$ for the LMS algorithm. The SNR value is 40dB and Doppler frequency is $f_D = 0.00005\text{rads}^{-1}$. The decision delay is set at $\Delta = L_{fb} - 1$, where L_{fb} is the length of the feedback tap. The tap lengths were set to the same value for both the feedforward and feedback filter. Based on the Figure, the RLS algorithm is less affected by the tap lengths than the LMS algorithm. For the RLS algorithm, the BER values are much lower and have less variations as the tap length is increased. For the LMS algorithm however, as the tap length increases the BER also increases noticeably. In the published work on the multivariate DFE, little explanation has been given about effects of tap lengths on the multivariate DFE, for example [2, 44, 38, 39]. The general values that were chosen were often numbers that are currently seen in the specifications, i.e. a tap length of 4, but appear to be chosen on an ad-hoc basis. Once again, simulation parameters such as λ, μ, f_D are the same as Section 5.2.2.

Figure 5.8 shows that the average BER for all channels increases as tap length increases. This coincides well with [19], since the adaptive algorithm performs better with single-tap filters. The Figure also shows that, on average, the RLS algorithms performs better than the LMS algorithm. However, later in Chapter 6 where pulse-shaping filters, synchronisation and a more complex modulation scheme are introduced, the adaptive algorithms do not work for low values of tap lengths. In general, a large number of taps for the feedforward and feedback filters do not directly contribute to the performance of the multivariate DFE as seen in [50, 49, 28]. In addition, increasing the number of taps drastically increases the complexity of hardware implementation, especially for the RLS algorithm which is discussed later in Chapter 7. Hence, a moderate number of taps are required and setting the tap lengths to 4 is acceptable for the purpose of implementation as well as producing reasonable results.

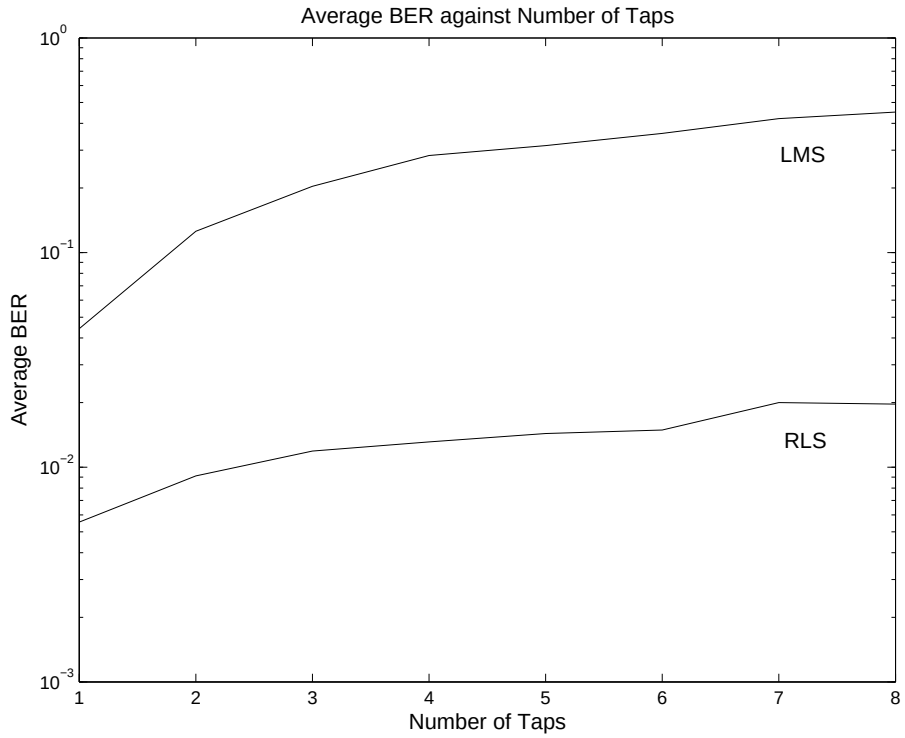

Figure 5.8 Average BER comparison between the LMS and RLS algorithm for different tap lengths

5.2.5 Doppler Frequency

One of the major variables in the simulation is the Doppler frequency. The ability of the simulation to change the Doppler frequency allows a comparison of the two adaptive algorithms in a mobile channel. The simulation uses the Jakes model to simulate this channel. A high Doppler frequency represents a rapidly changing channel and vice versa. The previous results were from simulations runs on a relatively static channel with a very low Doppler frequency. However, in this Section the performance of the two algorithms are shown when simulated through various degrees of channel variation. Figure 5.9 shows the channel variations for different Doppler frequency settings, f_D, in the simulation. The Figure shows 3 scenarios, a rapidly and moderately changing channel and a relatively static channel. An example of the typical error curves produced by a moderately changing channel can be seen in Figure 5.10. Evidently, both the LMS and RLS algorithm do not perform well for a moderately changing channel and this is also similar for a rapidly changing channel. Channel equalisation and tracking is difficult for moderately and rapidly changing channels and it can be concluded that this behaviour affects their performance drastically. The problem lies in tracking the channel after the training mode. Typically for adaptive algorithms on the DFE, once an error is made in the calculation, it can cause error propagation. As can be seen in Figure 5.10 for the RLS case, once the algorithm makes a mistake during the decision-

directed mode, it is difficult for the algorithm to return to its *correct* steady-state. In addition, as seen in Figure 5.10 the difference between the LMS and RLS algorithm for a moderately changing channel is stability during error propagation. Figure 5.10 shows that the LMS algorithm is less unstable than the RLS algorithm when the algorithm makes a decision error. The fast converging nature of the RLS algorithm has its drawbacks, meaning that the algorithm can converge and diverge very quickly. This leads to a higher BER even though the RLS algorithm performed well for the majority of the packet. One thing to note is that the RLS algorithm converges very well during training mode. Hence, a good suggestion is to convert to using the LMS algorithm for tracking the channel, thus using a hybrid mode during channel equalisation. Figure 5.11 shows the average BER for all channels against Doppler frequency. The variations in the channel over time results in changes in the optimal tap weights. This causes the algorithm to continually change its weights and not reach a steady-state value as in Figure 5.10. The difficulty in tracking the channel, often results in poor performance and therefore higher BER. Figure 5.11 shows that as the Doppler frequency increases, average BER for all channels increases. The Figure also shows that the RLS algorithm outperforms the LMS algorithm, especially when the channel is relatively static. The simulation parameters such as μ, λ, SNR and tap length are the same as Section 5.2.2.

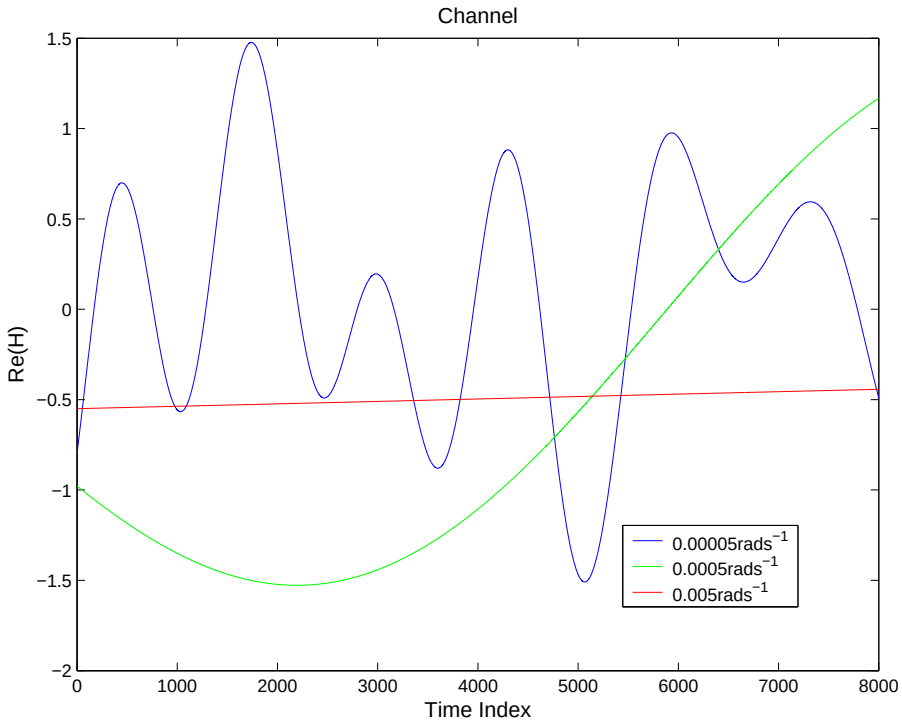

Figure 5.9 Examples of the temporal behaviour of the channel coefficient for different Doppler frequencies

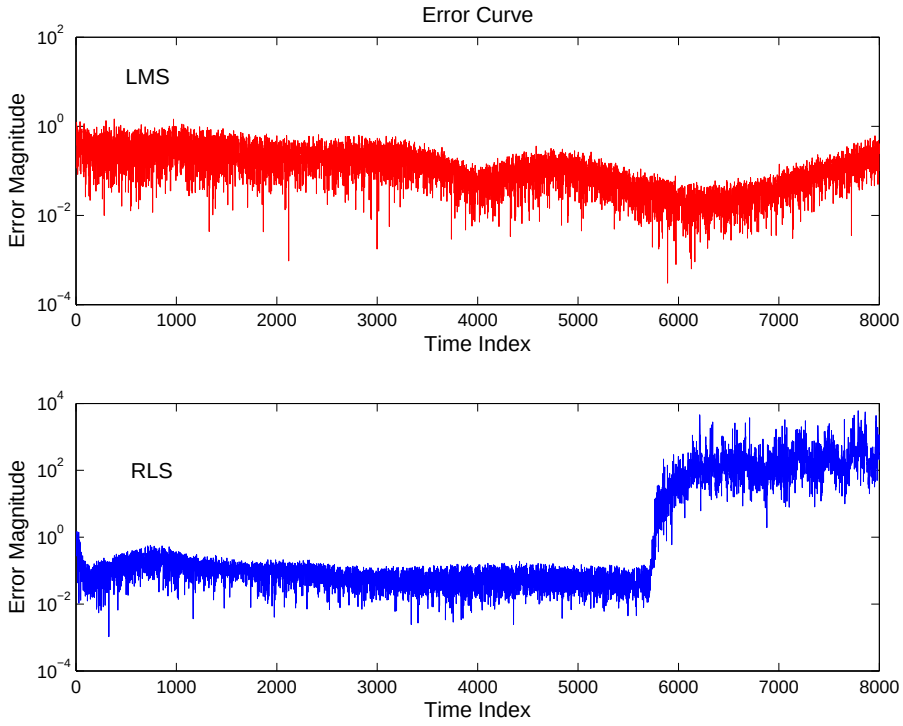

Figure 5.10 Error curve showinng the difference between the LMS and RLS algorithms for a moderately changing channel, $f_D = 0.0005 rads^{-1}$

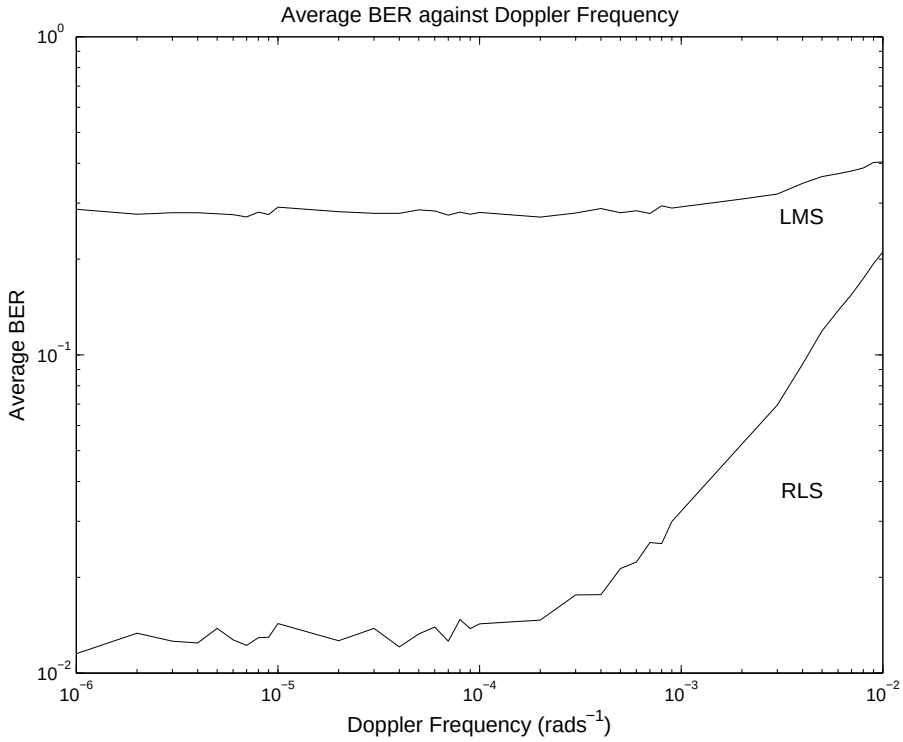

Figure 5.11 BER for different values of Doppler Frequency

5.2.6 Decision Delay

Decision delay has a profound effect on the stability of the RLS algorithm. Hence, we investigate its performance in this Section. Simulation parameters such as μ, λ, f_0, SNR and tap lengths are the same as Section 5.2.2. In Figure 5.12, we show the relationship between decision delay and average BER for all channels. For these results, the average BER for all channels does not vary greatly with changes in decision delay. The only noticeable difference is that the average BER for the RLS algorithm is lower than that of the LMS algorithm.

The major concern regarding decision delay in the adaptive multivariate DFE is the effect it has when the RLS algorithm is implemented on it. From Chapter 3, the RLS algorithm requires several variables such as $\mathbf{K}_p(k)$, $\mathbf{P}(k)$ and $R_e(k)$ for calculating the updated weights for each iteration as seen in Section 3.4.4. During the simulation runs, the maximum value of the $\mathbf{P}(k)$ matrix was observed and showed divergence when using the structure of the multivariate DFE proposed in [2]. When decision delay was implemented in this structure, the divergence subsided. Figure 5.13 shows the plot of a diverging and converging maximum $\mathbf{P}(k)$ entry. By the maximum entry we mean the maximum absolute value of all entries in the $\mathbf{P}(k)$. For ease of notation, we simply refer to this as the maximum $\mathbf{P}(k)$ entry in the rest of this thesis. The plot shows that the multivariate DFE produces converging $\mathbf{P}(k)$ matrices when having decision delays equivalent to $\Delta = L_{fb} - 1$, where L_{fb} represents the feedback tap length. In this case, $L_{fb} = 4$ and therefore $\mathbf{P}(k)$ converged for $\Delta = 3$. The selection of the decision delay as $\Delta = L_{fb} - 1$ is consistent with work on the selection of decision delays for DFEs [81].

From Figure 5.12, there are hardly any differences when simulating the RLS algorithm for different values of the decision delays. However, a major difference can be observed in the $\mathbf{P}(k)$ matrix. The particular example shown in Figure 5.13 illustrates that $\mathbf{P}(k)$ shows convergence when $\Delta = 3$, i.e. when $\Delta = L_{fb} - 1$. Smaller values of Δ surprisingly, do not show any form of convergence for $\mathbf{P}(k)$. Further analysis shows that this is related to a property of $\mathbf{P}(k)$ discussed in Section 5.2.8.

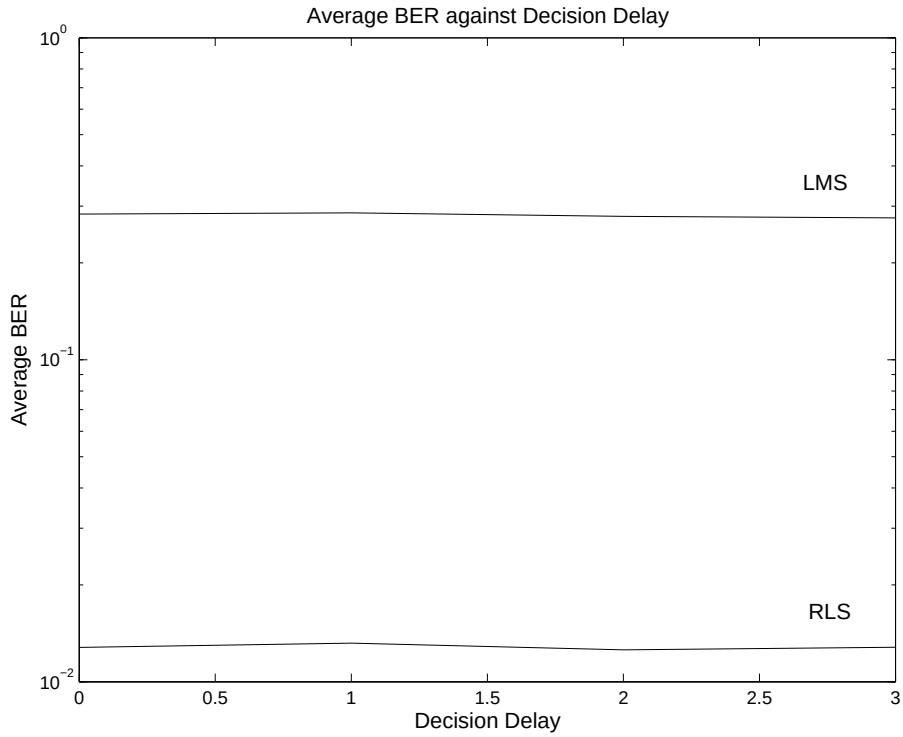

Figure 5.12 Average BER for different decision delays for the multivariate DFE for both the LMS and RLS algorithms

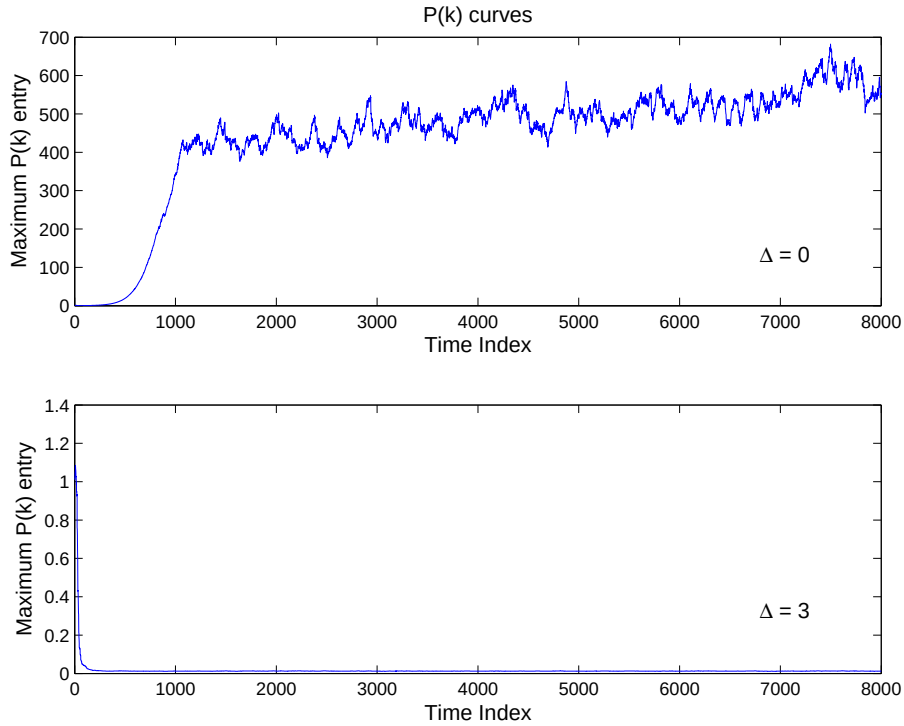

Figure 5.13 Effect of decision delay on the maximum $\mathbf{P}(k)$ entry

5.2.7 Forgetting Factor

One of the most notable observations from the simulations is the effect of the forgetting factor on the RLS algorithm implementation in the multivariate DFE. Figure 5.14 shows the error curve for the 1^{st} channel for various forgetting factor values, $\lambda \epsilon \{0.95, 0.96, 0.97, 0.98, 0.99, 1\}$. The simulation parameters such as μ, tap lengths, f_D and SNR are the same as Section 5.2.2. The packet length is 8000 and the training length is 500. The purpose of the forgetting factor is to slowly *forget* the past data so that the more recent observations will eventually dominate. From Figure 5.14, the RLS algorithm performs very well especially during training mode for all values of λ between 0.95 and 1, which clearly illustrates the fast convergence of the RLS algorithm. However, for $\lambda = 1$ the algorithm does not perform as well as the other λ values as it enters decision-directed mode. The RLS algorithm with $\lambda = 1$ does not track the channel very well. By observing the error curve in Figure 5.14, the curve diverges away from the error floor. This undesirable property can be explained through the purpose of the forgetting factor. By setting the forgetting factor λ to 1, the algorithm does not forget the past data. This basically means that the algorithm has an infinite memory of the channel characteristics. Based on [19], the use of a forgetting factor in general is to ensure that data in the distant past are *forgotten* in order to afford the possibility of following the statistical variations of the observable data when the DFE operates in a nonstationary environment. Therefore, for $\lambda < 1$ the algorithm performs at its best because during decision-directed mode, the algorithm tracks the channel by forgetting past channel characteristics. For implementation, a RLS algorithm with $\lambda = 1$ is more desirable than $\lambda < 1$ because the hardware device requires no multipliers at all. By referring to the equations shown in Section 3.4.4, the RLS algorithm would require 3 more multipliers for any $\lambda < 1$. The forgetting factor also affects the $\mathbf{P}(k)$ matrix especially using the initial multivariate DFE structure [2] that does not utilise decision delays ($\Delta = 0$). Figure 5.15 shows two scenarios where the $\mathbf{P}(k)$ converges for $\lambda = 1$ and diverges for any $\lambda < 1$. Note that the size of the $\mathbf{P}(k)$ matrix is 32 by 32.

The simulation results show that adjusting the forgetting factor below 1 affects $\mathbf{P}(k)$ if the multivariate DFE structure does not incorporate a decision delay. Incorporating the decision delay, mitigates the problem with $\mathbf{P}(k)$ as shown in Figure 5.15. Hence, $\lambda = 1$ gives poor error performance but a stable $\mathbf{P}(k)$ matrix when $\Delta = 0$. For $\Delta = 3$, $\lambda < 1$ is satisfactory both for error and $\mathbf{P}(k)$ stability. There is clearly an interaction between parameter values which becomes very important in implementation. Such issues are not well documented and straightforward implementation of RLS algorithms in the literature can result in very poor performance.

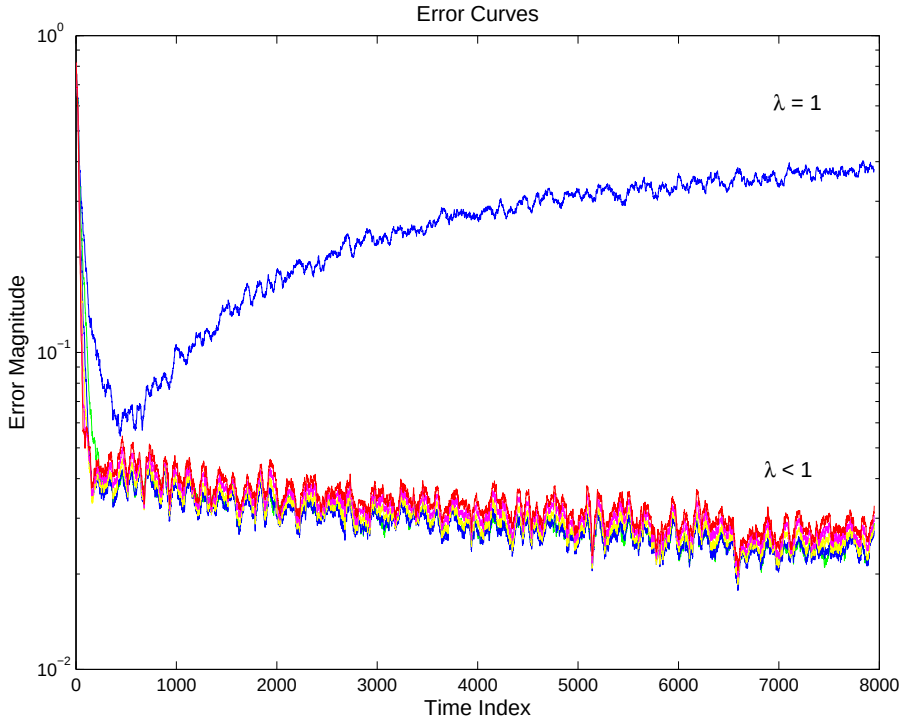

Figure 5.14 Moving average of the error curves for the RLS algorithm with different forgetting factors, $\lambda\epsilon\{0.95, 0.96, 0.97, 0.98, 0.99, 1\}$

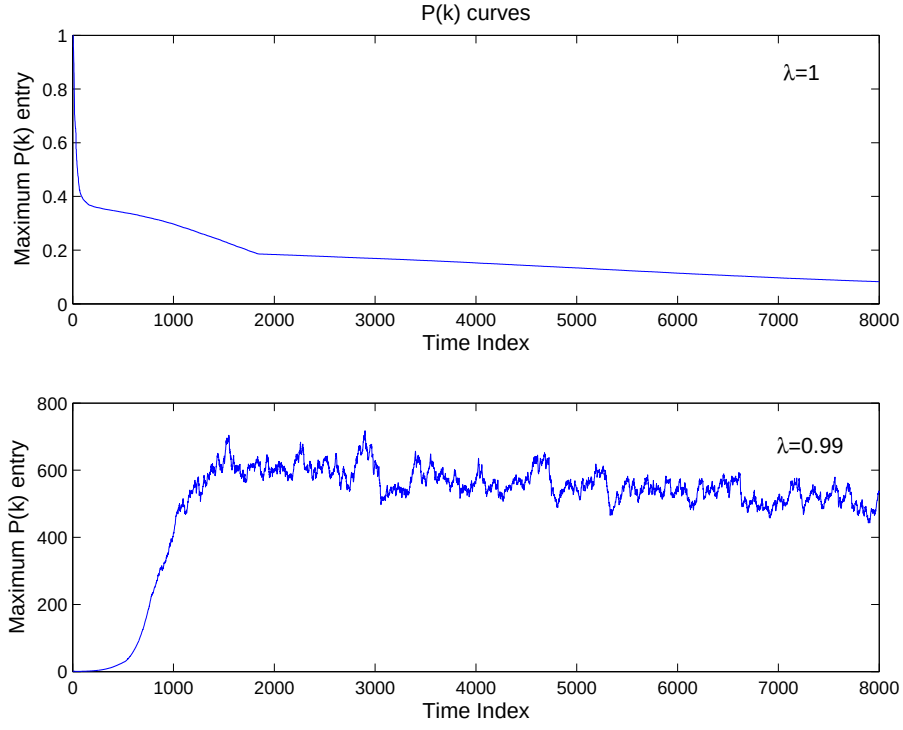

Figure 5.15 The maximum $\mathbf{P}(k)$ values in the RLS algorithm with different forgetting factors, $\lambda = 0.99$ and $\lambda = 1$

5.2.8 Inverse Correlation Matrix

The inverse correlation matrix, $\mathbf{P}(k)$, encountered major instability issues during the RLS algorithm multivariate DFE simulation. As illustrated in the earlier Sections, $\mathbf{P}(k)$ diverged if there was no decision delay. Furthermore, particular values of decision delay ($\Delta \geq 3$) were required to ensure that $\mathbf{P}(k)$ converged as seen in Figure 5.13. The initial structure of the multivariate DFE in the Tait platform did not incorporate any decision delays because it initially dealt with a simple LMS algorithm. Furthermore, no published work has pointed out difficulties with a zero delay and a zero delay is used in some published algorithms [2, 43, 44].

The zero delay setting caused instability in the following way. Consider, for simplicity, a SISO system with 2 feedforward taps and 2 feedback taps. For such a system the input vector is $\mathbf{u}(k) = \begin{bmatrix} y_1(k) & y_1(k-1) & -\hat{x}_1(k) & -\hat{x}_1(k-1) \end{bmatrix}$. With zero delay, the feedback element $-\hat{x}_1(k)$ contains the estimate of the signal just detected. This signal is contained in the input $y_1(k-1)$. Hence, $-\hat{x}_1(k)$ and $y_1(k-1)$ are expected to be highly correlated at reasonably low BERs, since $\hat{x}_1(k)$ is the estimate of the signal carried by $y_1(k-1)$. In a simulation run it was found that the correlation coefficient between $\hat{x}_1(k)$ and $y_1(k-1)$ was 98%. Hence, the correlation matrix $\Phi(k)$ will be nearly singular as the two variables are extremely strongly related. As a result, $\Phi(k)$ will have at least one very small eigenvalue and $\mathbf{P}(k) = \Phi^{-1}(k)$ will have a tendency to become large. Hence, divergence has been traced to the properties of the eigenvalues of $\Phi(k)$. The near singularity of $\Phi(k)$ will also have an impact on the numerical stability of the recursive calculations, possibly leading to a non-symmetric, non-positive definite $\mathbf{P}(k)$ matrix.

The problems with the diverging $\mathbf{P}(k)$ matrix are the implementation problems that come with it. Divergence causes the range of fixed point numbers to increase unnecessarily. There is a simple solution to mitigate this divergence, and this was by selecting a forgetting factor $\lambda = 1$ as seen in Figure 5.15. Other methods include saturating $\mathbf{P}(k)$ or resetting the value of $\mathbf{P}(k)$. From observation, saturating $\mathbf{P}(k)$ by setting a maximum limit on the number may cause error propagation resulting in divergence in the error curve. Further details of this phenomenon are given in Section 6.2.4. Resetting $\mathbf{P}(k)$ to its initial value, the identity matrix is another technique that might be reasonable. This technique is commonly used in some adaptive control theory but is not mentioned in any literature regarding equalisation with the RLS algorithm [63, 64]. Finally, using $\lambda = 1$ is not desirable, since this implies an infinite memory and an inability to adapt to changing channel conditions. In general, the solution we have adopted is to implement decision delays in the multivariate DFE used in the Tait platform. If the decision delays do not solve this diverging maximum $\mathbf{P}(k)$ problem in fixed point implementation, the two methods, namely saturation or resetting the matrix, can be used. These issues are discussed further in Chapter 6.

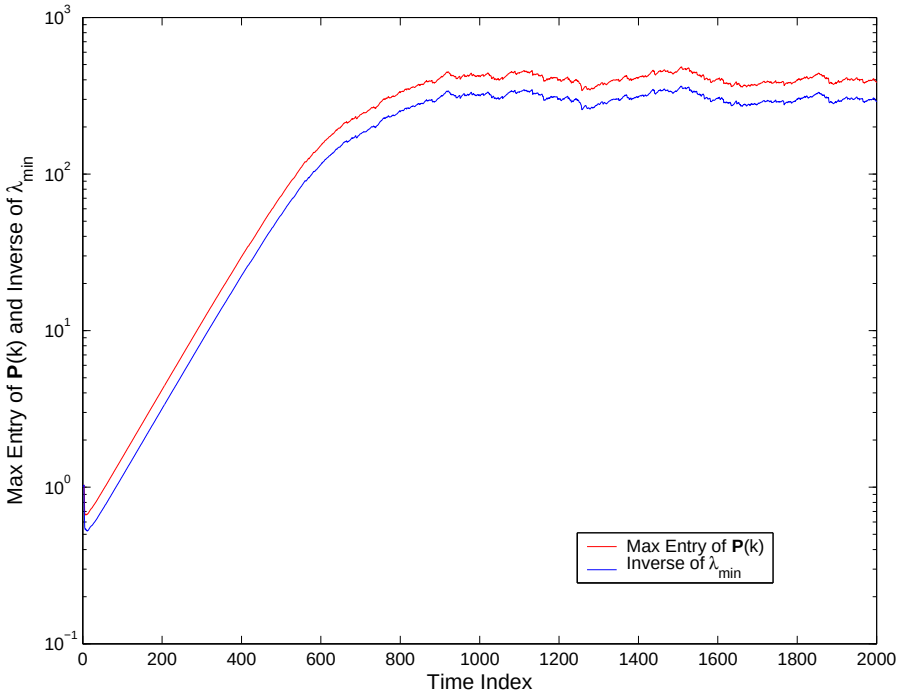

Figure 5.16 Relationship between the maximum $\mathbf{P}(k)$ entry and the minimum eigenvalues of Φ, λ_{min}

5.3 DISCUSSION

The intention of Group Research was to study various adaptive filter algorithms that can be implemented on the Tait platform. This involves investigating the performance of these algorithms on the multivariate DFE, with a focus on real-time implementation. Unlike most equalisation work, the adaptive algorithms have to be implemented on the Tait platform which has several unique features which were described in Section 4.

The base simulations provide a simple, uncluttered environment for adaptive algorithm study on the multivariate DFE. The Tait platform has components such as $\pi/4$-DQPSK modulation and synchronisation bits which were not included in the base simulations. As can be seen, intermediate values like the $\mathbf{P}(k)$ matrix can cause instability because of the range of values that $\mathbf{P}(k)$ can take. The larger the range in a fixed point environment, the less accurate the number will be. The simulations illustrated various issues including decision delay, the effects of hybrid mode, Doppler frequency, etc.

Results show that the initial proposed RLS algorithm and the structure of the multivariate DFE shown in [2] produce undesirable results. The proposed RLS algorithm was inherently unstable due to the structure of the multivariate DFE which did not have a decision delay. The simulations here considered methods to avoid these complications. Analysis shows that the RLS algorithm must utilise decision delays to ensure that $\mathbf{P}(k)$ converges and shows that the RLS algorithm outperforms the LMS algo-

rithm. The results also show that practical issues and suitable parameter settings for the multivariate DFE are far from resolved. The literature has relatively little guidance on these issues, beyond adhoc choices of parameter values. Furthermore, the parameters show interactions so that, for example, a sensible setting of $\lambda < 1$ for $\Delta = 3$ does not work for $\Delta = 0$. This is of some concern, since the algorithm must be capable of reasonable performance in a very broad range of environments for it to be useful. Sensitivity to the parameter makes this robustness questionable.

Chapter 6

TAIT PLATFORM PERFORMANCE INVESTIGATION AND SIMULATION

6.1 INTRODUCTION

This Section describes the overall simulation and investigation into the feasibility and performance of the RLS algorithm on the Tait platform. The Section includes a brief description of the simulation system and results from the two stages of simulation, floating and fixed point. The Chapter also investigates the issue with the $\mathbf{P}(k)$ matrix and applies divergence mitigation techniques. Finally, real-time received data are used to evaluate the adaptive algorithms on the multivariate DFE.

This Chapter presents the results from the Tait platform simulations which differ from the base simulations as seen previously in Chapter 5. The adaptive multivariate DFE structure is identical to the base simulation but in this Chapter we incorporate specifications from the Tait platform for example modulation, synchronisation, pulse-shaping filters and a different channel model. The specifications were shown previously in Table 4.1. A summary of the differences between the two simulations are shown in Table 4.5. In general, the Chapter is divided into three parts as shown below:

1. Floating point simulation

2. Fixed point simulation

3. Fixed point simulation using real-time received data

Extensive simulation runs were performed for this Chapter. Representative results are presented but for reasons of space the majority of the graphs are omitted. However, the conclusions drawn in this Chapter are supported by the majority of the simulations as well as the results shown.

6.2 FLOATING POINT SIMULATION

6.2.1 Overview

The floating point simulation is a flexible system allowing various modifications for real-time implementation of the adaptive multivariate DFE. It incorporates the specifications from Tait platform as seen in Table 4.1, in particular the synchroniser, modulation, pulse-shaping filters and packet structure. The channel model used in the simulation is the quasi-stationary model as defined in Section 4.6.2. This model is the initial simulation channel model used to investigate the LMS algorithm and is therefore used in the RLS algorithm simulations. The model is suitable because the real-time platforms are stationary and are tested within an indoor environment. In addition, this allows more focus on observation of the adaptive algorithm performance on the multivariate DFE.

The simulation system begins with the initialisation of the adaptive multivariate DFE system. The packet structure is initialised which includes training, synchronisation and payload data. The data is $\pi/4$-DQPSK modulated, pulse-shaped and sent through the quasi-stationary channel. The received data gets fed to a synchroniser that detects the synchronisation bits in the packets. The index of the synchronisation bits allow the multivariate DFE to detect the start of the packet for processing and adaptive equalisation. More information regarding the Tait platform can be seen in Chapter 4.

6.2.2 Equalisation Results

From all the analysis and theory considered thus far, the RLS algorithm has always been superior in performance. The base simulations obviously show that the RLS algorithm converges faster which results in a lower average BER than the LMS algorithm. The next step is the evaluation of performance of the LMS and RLS algorithms on the Tait platform with more focus on the RLS algorithm. The next few Figures will illustrate the outputs generated by the simulations. Each Figure is arranged as $\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}$ to represent the 4 streams. As mentioned before, the Tait specifications use 4 feedforward and feedback taps, $\mu = 0.0625$ and $\lambda = 1$. Training length is set at 448 samples and synchronisation length is set at 64 samples. The decision delay Δ was set to 3 as it was the optimum delay setting as seen in Section 5.2.6.

Figure 6.1 shows that the error curves for both adaptive algorithms converge very well. Figures 6.2 and 6.3 show the decoded soft outputs from the multivariate DFE. The soft equalised outputs which show the 8 points are not given because the decoded soft outputs are generated from these values. By observing the Figures, once again it is clear that the RLS algorithm outperforms the LMS algorithm in the floating point simulation. The error curve converges to the error floor faster than the LMS algorithm. The decoded soft outputs are less noisy and more well-defined for the RLS algorithm.

Another notable observation is the effect of the synchronisation. From Figure 6.1, the convergence is smooth and uninterrupted by the brief calculation of the magnitude for packet detection.

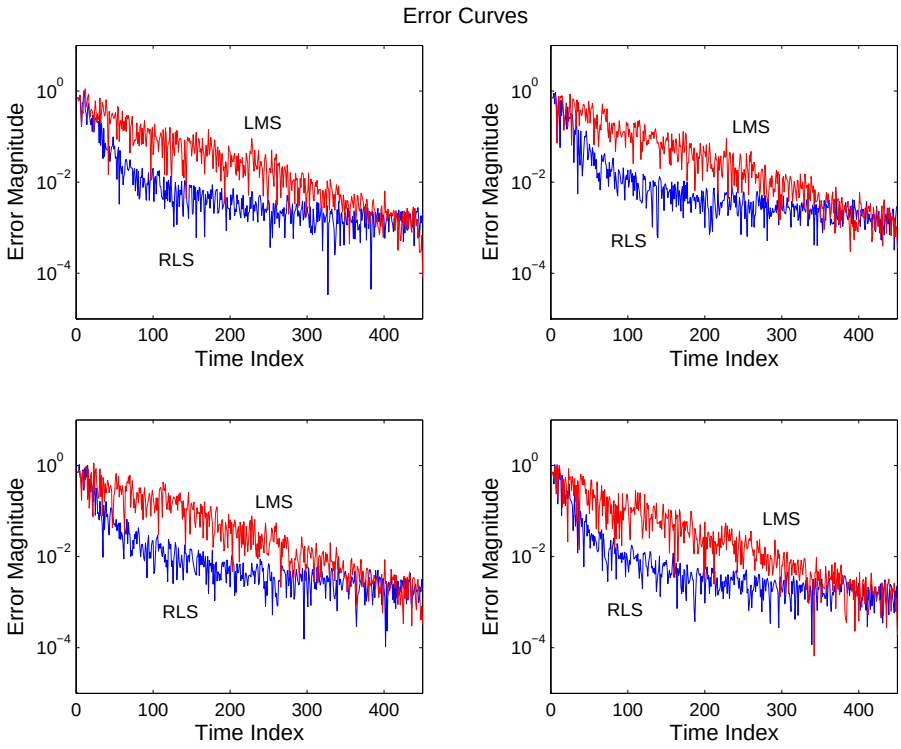

Figure 6.1 Error curves for for the LMS and RLS algorithm

Based on the results shown, both LMS and RLS algorithms have relatively good performance in terms of error convergence, and their decoded soft output resembles the desired QPSK constellation. The specifications set for the adaptive multivariate system are suitable for implementation on the FPGA. However, there are still issues with the RLS algorithm as mentioned earlier in the background theory and base simulation results. The issue is basically the diverging $\mathbf{P}(k)$ matrix when the forgetting factor λ is below 1. This was shown earlier in Section 5.2.8 and is discussed for the Tait simulations case in Section 6.2.4.

6.2.3 Filter Order

The base simulations showed that the adaptive multivariate DFE worked for all filter tap lengths. A similar study is done on the floating point Tait simulations. The simulation showed that the DFE cannot produce any useful results below a certain tap length. Figures 6.4 and 6.5 show the outputs for simulations with tap lengths of 2 and 4. Figure 6.4 shows the error curves for tap lengths of 2 and 4 for a multivariate DFE using the RLS algorithm. The curve that is not converging corresponds to the

DFE with a tap length of 2. Evidently when the error does not converge, the results produced by the DFE are useless as seen in Figure 6.5. The decoded soft output cannot produce a QPSK-like constellation and detection has completely broken down. Note that these results are similar to those for the LMS algorithm, as well as for a DFE tap length of 1 for both algorithms. It is evident that for any tap length below 3, the multivariate DFE ceases to produce any useful results for both the LMS and RLS algorithm. This means that the minimum tap length for the multivariate DFE is 3. These observations are not the same as for the base simulations as seen in Chapter 5. Most likely, this can be related to the complexities of the Tait simulations such as pulse-shaping filters, synchronisation and modulation.

6.2.4 Instability

Overview

This Section describes instability issues in the RLS algorithm implementation on the multivariate DFE. From Section 6.2.2, output results produced by the adaptive multivariate DFE are good. Selecting the tap length above 3 ensures that the multivariate DFE is working as seen in Section 6.2.3. However, intermediate parameters have not been observed for the RLS algorithm implementation and they are discussed in this

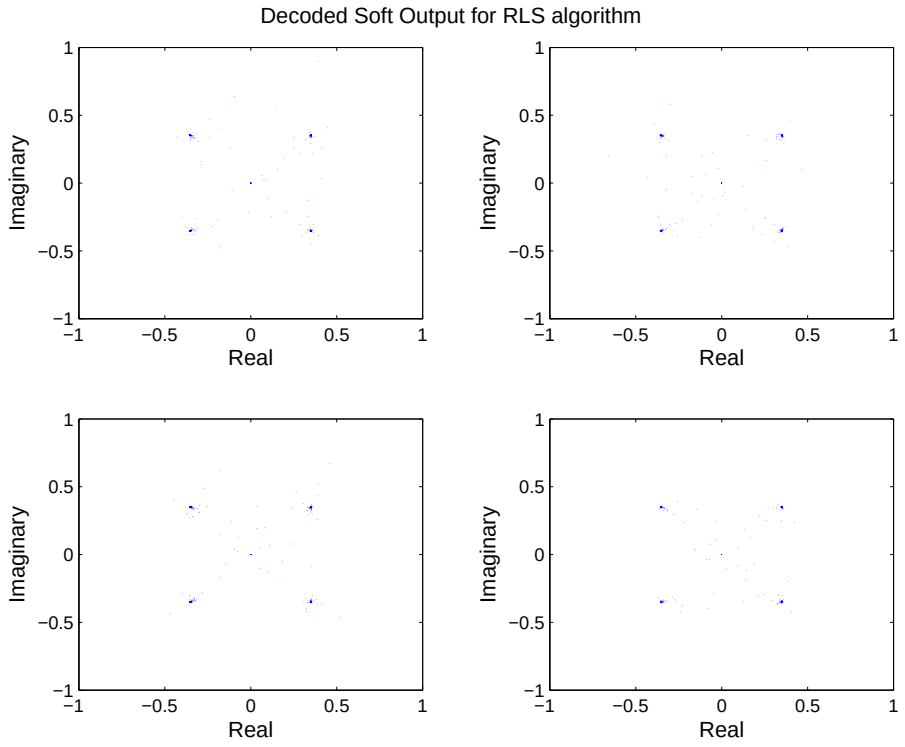

Figure 6.2 The decoded soft output for the RLS algorithm

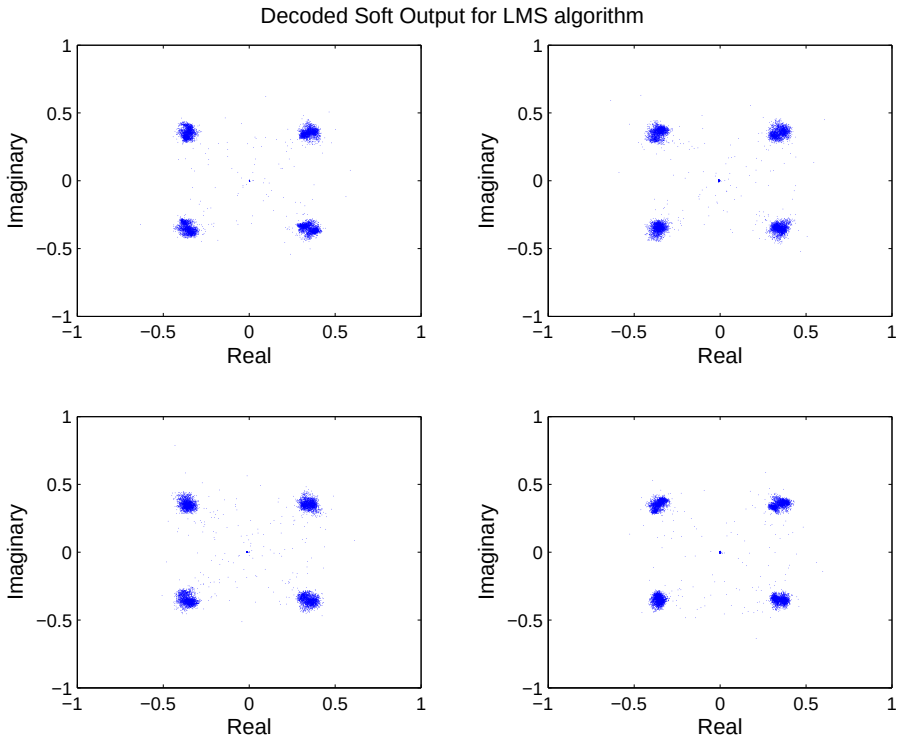

Figure 6.3 The decoded soft output for the LMS algorithm

Section. A few issues are described briefly, in particular the divergence of the $\mathbf{P}(k)$ matrix. Mitigation of this instability is also described.

Filter Order

Section 6.2.3 described one issue that affects the performance of the adaptive multivariate DFE namely, filter order. If the tap length is of reasonable length, set at 4 in the specifications, the results for both adaptive algorithms are acceptable. The multivariate DFE also works for a tap length of 3 but no less. The only major difference is that the RLS algorithm outperforms the LMS algorithm based on error curves and decoded soft output results. However, instability is still an issue with these adaptive algorithms. For example, performance of the LMS algorithm is affected by its step-size parameter μ. If μ is too large, instability occurs. Fortunately, this problem can be easily mitigated by selecting μ such that it exhibits stability.

Forgetting Factor and Decision Delay

As seen in Section 5.2.8 of the previous Chapter, the most prevalent instability issue during the base simulation is the diverging $\mathbf{P}(k)$ matrix for the RLS algorithm implementation on the multivariate DFE. The divergence is related to the forgetting

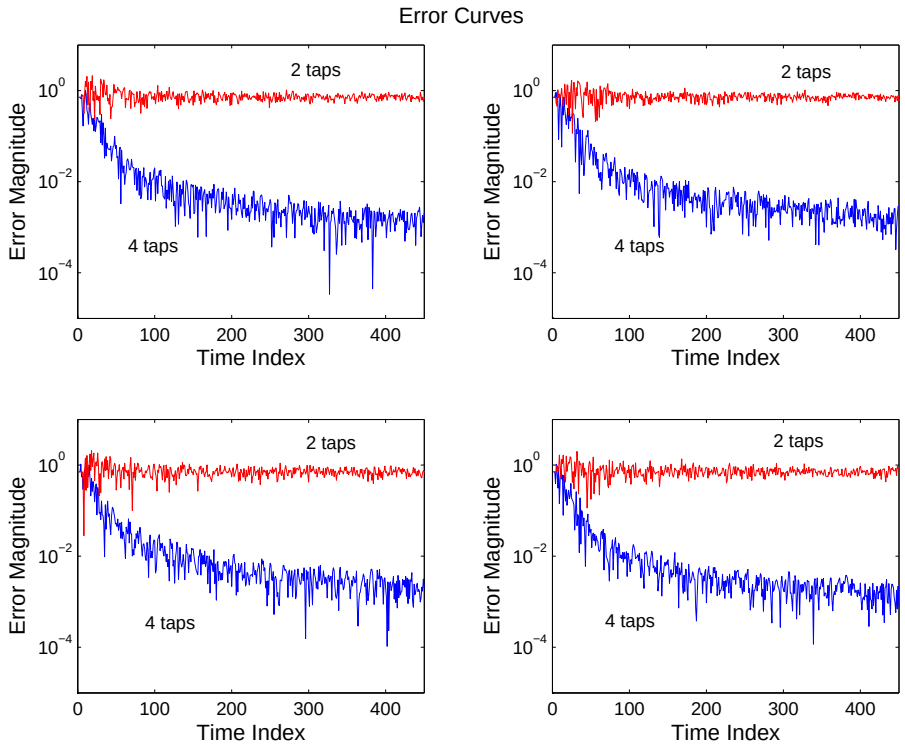

Figure 6.4 The error curves for each channel using the RLS algorithm with a tap length of 2 and 4

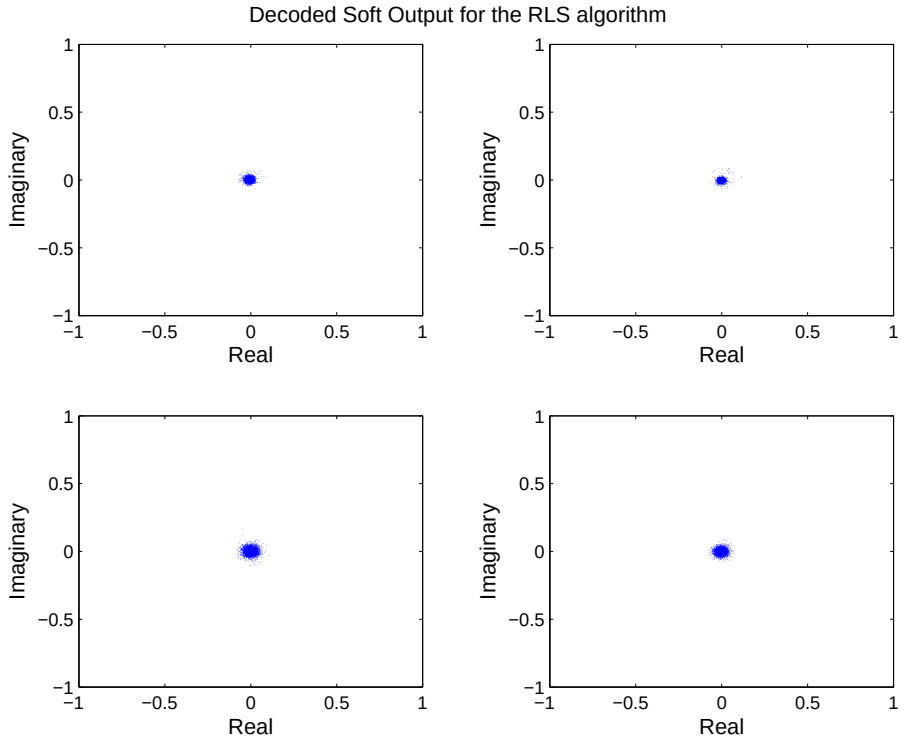

Figure 6.5 The decoded soft output for each channel using the RLS algorithm with a tap length of 2

factor λ such that the intermediate RLS parameter, $\mathbf{P}(k)$ diverged for λ less than 1. It was found that the divergence can be mitigated by implementing decision delay in the DFE. Divergence of $\mathbf{P}(k)$ also occured for the floating point Tait simulations. Shown in Figures 6.6 and 6.7 are curves showing examples of converging and diverging $\mathbf{P}(k)$ matrices respectively. The matrix only converges for a forgetting factor λ of 1, otherwise it diverged for any value below 1. It it worth noting that in floating point the

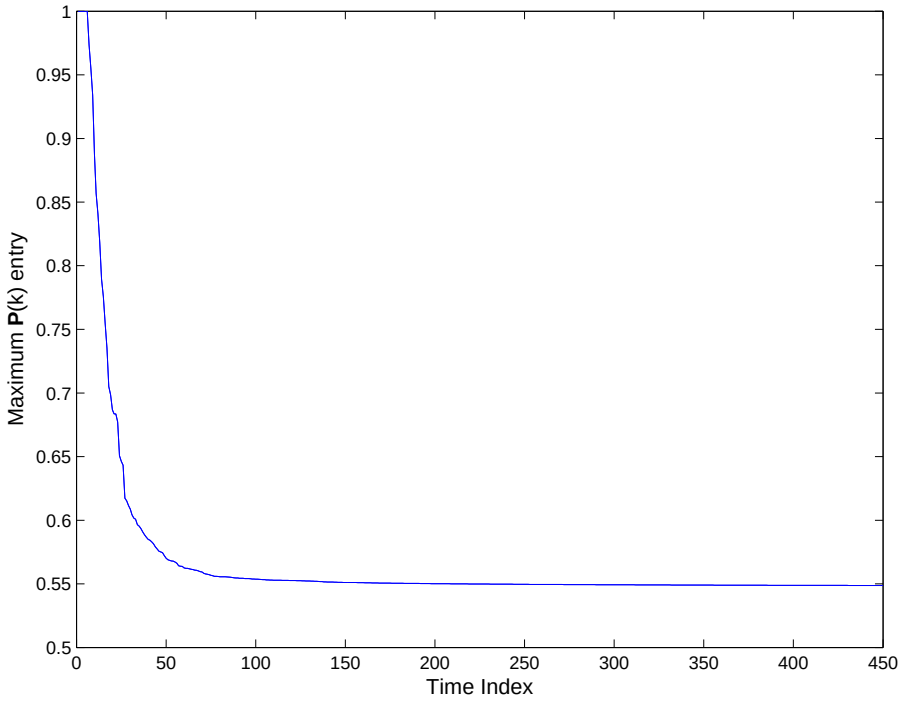

Figure 6.6 Convergence of the $\mathbf{P}(k)$ matrix for the RLS algorithm where $\lambda = 1$.

multivariate DFEs are still working, producing converging error curves and the 4 scattered points as its decoded soft output as seen in Section 6.2.2. However, this successful performance is due to the floating point ability to handle the large values in the $\mathbf{P}(k)$ matrix. It was found in Chapter 5 that decision delays can help the $\mathbf{P}(k)$ matrix to converge. Unfortunately, implementing the decision delay did not prevent the elements of $\mathbf{P}(k)$ from diverging in the Tait simulations. Most likely, the complexities in the system caused this difference. Hence, the RLS algorithm implemented on the DFE can only be implemented if the forgetting factor $\lambda = 1$ unless other instability mitigation methods are employed. In the next Section, the effects of saturation are illustrated when the $\mathbf{P}(k)$ matrix diverges.

Saturation

In hardware, saturation is a common occurrence because of fixed point arithmetic. This is discussed because the RLS algorithm is implemented on the FPGA which requires

the number to be in fixed point. Fixed point arithmetic only allows a limited range of numbers. Saturation occurs when the number exceeds its range and thus saturates to the maximum or minimum based on the precision of the fixed point arithmetic. For example, for n-bit fixed point arithmetic using two's complement numbers, the range of the number is limited from -2^{-n+1} to $2^{n-1} - 1$. Using the specification seen in Table 4.1, the precision used is 16-bits. Therefore, the range of numbers available is from -32768 to 32767. This allows a higher resolution in fixed point. The effect of saturation can be observed in floating point simply by preventing numbers from increasing excessively. As we know, the elements of the $\mathbf{P}(k)$ matrix have a tendency to diverge for $\lambda < 1$. Other RLS parameters did not exhibit this behaviour, converging to a certain value within a reasonable range. During the Tait simulations, elements of the $\mathbf{P}(k)$ matrix were clipped to a maximum of $4+j4$ and a minimum of $-4-j4$ when it started to diverge. Figure 6.8 illustrates this saturation effect on the matrix. Evidently, this affects the overall performance of the system by observing outputs from the DFE. Figure 6.9 shows the error curves when $\mathbf{P}(k)$ is clipped. Evidently, clipping has resulted in instability since the error curves are diverging. The effect of saturation is that the $\mathbf{P}(k)$ matrix has lost its symmetric and strongly diagonal nature. Figures 6.10 and 6.11 illustrate these matrix properties via a contour plot for two scenarios. Figure 6.10 shows a symmetric, strongly diagonal matrix resulting from normal simulations without clipping. Figure 6.11 shows a non-symmetric matrix when saturation is applied, namely

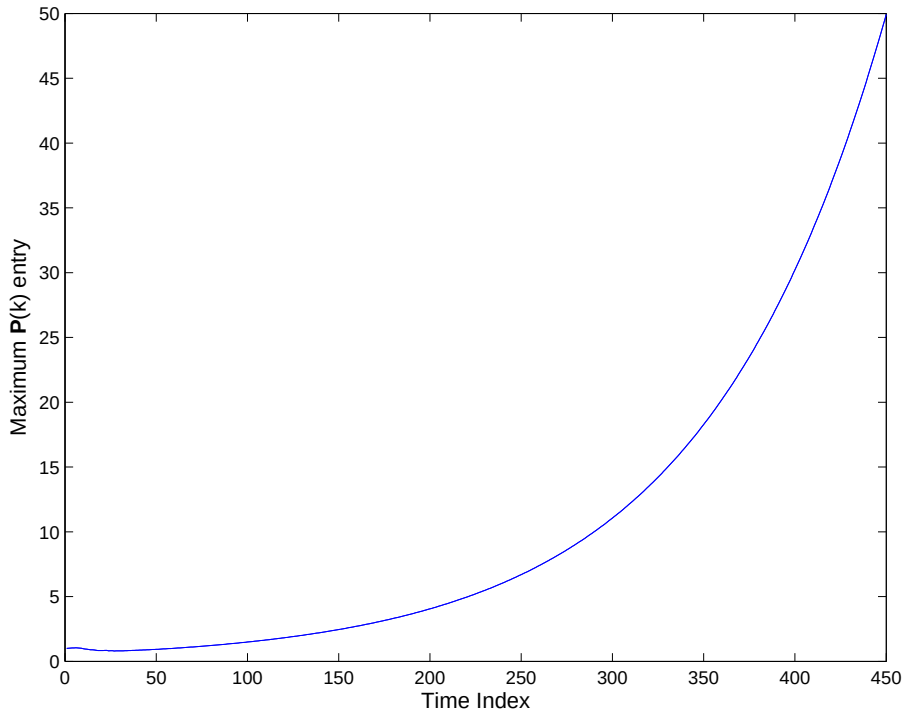

Figure 6.7 Divergence of the $\mathbf{P}(k)$ matrix for the RLS algorithm where $\lambda = 0.99$

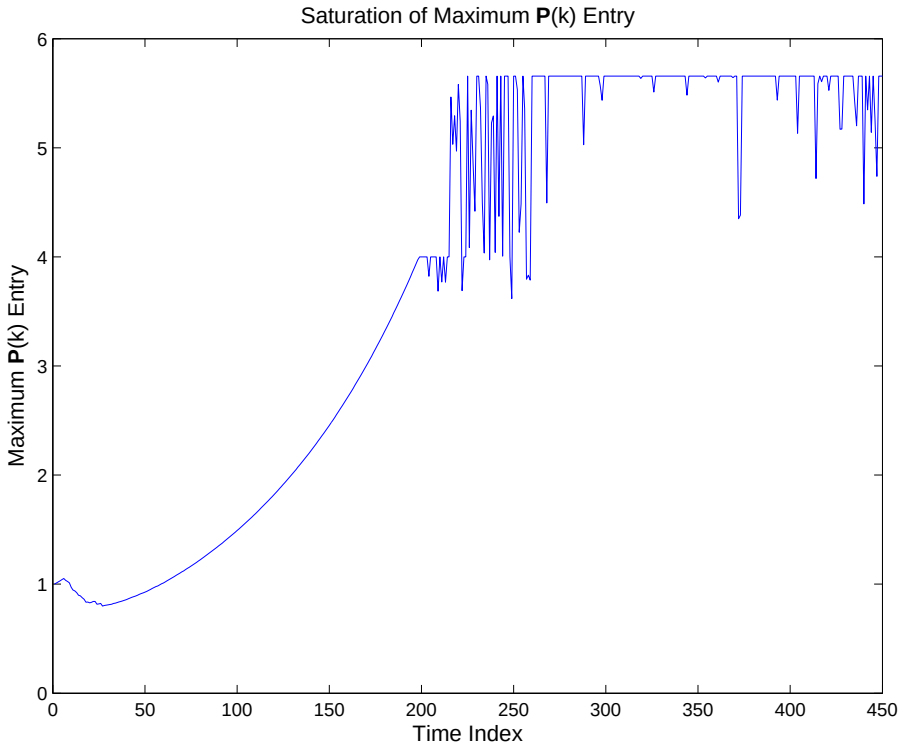

Figure 6.8 Effects of clipping the maximum entries of $\mathbf{P}(k)$ matrix when $\lambda = 0.99$. Shown here are absolute values of the element.

to a maximum of $4 + j4$ and a minimum of $-4 - j4$. The next Section describes the alternative method of resetting that maintains these desirable properties as discussed in Section 5.2.8.

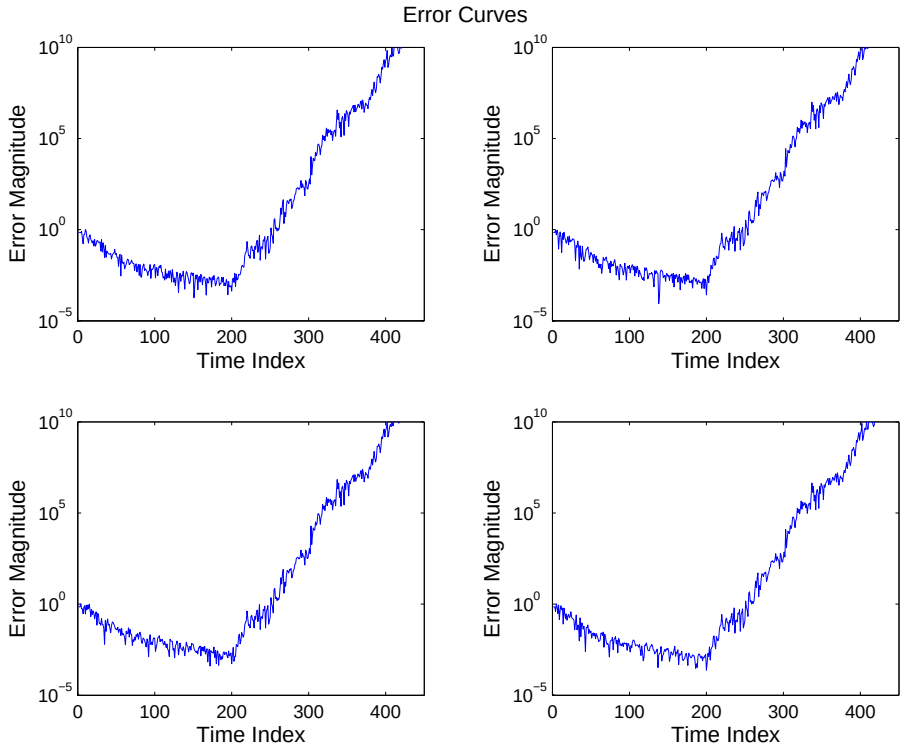

Figure 6.9 Effects of clipping the maximum value of $\mathbf{P}(k)$ matrix on the error curve.

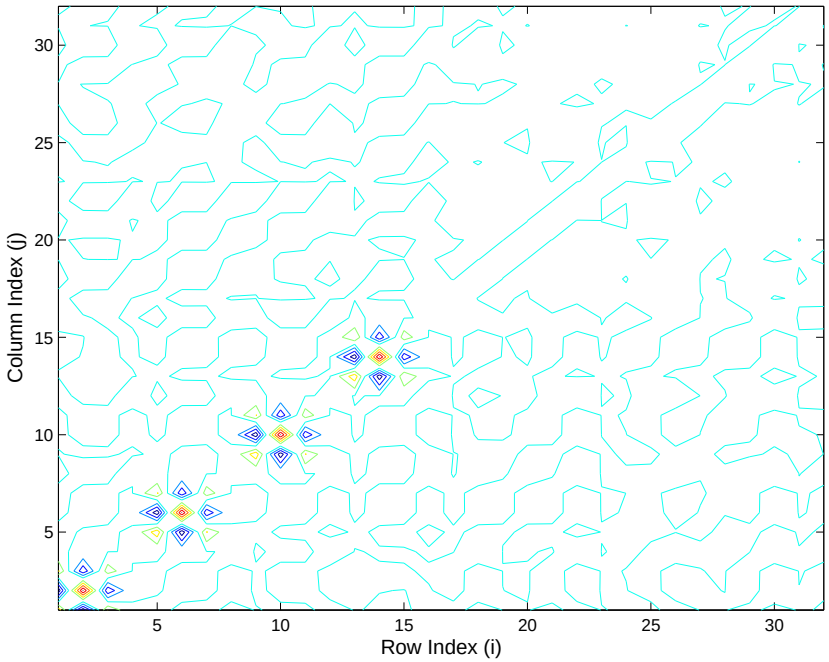

Figure 6.10 Contour plot of the $\mathbf{P}(k)$ matrix where (i, j) represents the $(i, j)^{th}$ element of the matrix. The $\mathbf{P}(k)$ matrix exhibits approximate symmetry without saturation.

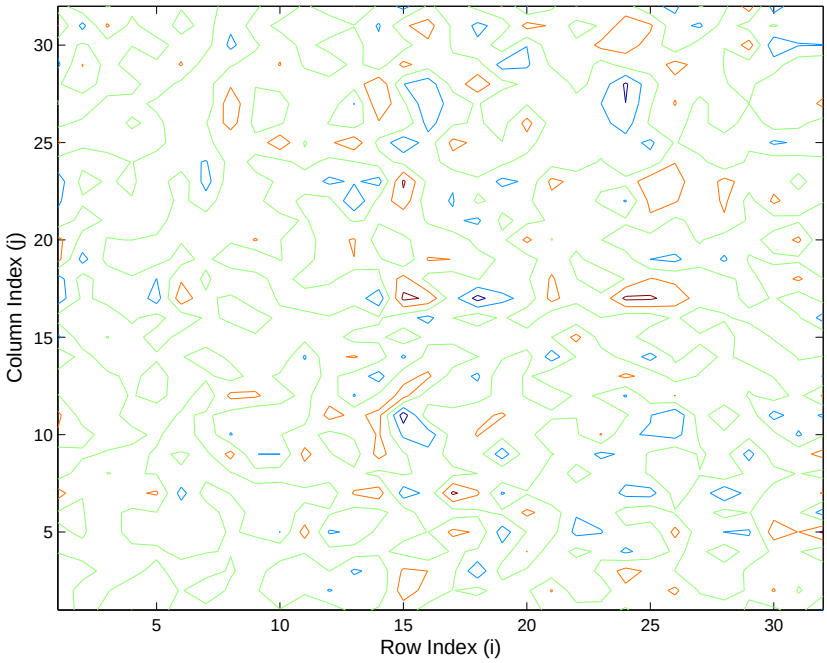

Figure 6.11 Contour plot of the $\mathbf{P}(k)$ matrix where (i, j) represents the $(i, j)^{th}$ element of the matrix. The $\mathbf{P}(k)$ matrix exhibits marked non-symmetry with saturation.

Reset Technique

As seen in Section 6.2.4, the effects of saturation causes the DFE to produce unstable results. This prompted a new method to remove this undesirable property of the $\mathbf{P}(k)$ matrix. From the literature survey, various methods have been proposed for removing saturation effects of the $\mathbf{P}(k)$ matrix when the RLS algorithm is applied to adaptive control systems. One such method involves resetting the $\mathbf{P}(k)$ matrix whenever the value diverges [63, 64]. The $\mathbf{P}(k)$ matrix resets to its original initial value which is $\delta^{-1}\mathbf{I}$ where δ is a small number. For the Tait simulation as well as the RLS algorithm on the multivariate DFE, δ is 1. Figure 6.12 illustrates the error curves produced by this method. The error curves exhibit convergence. From observation, this is a valid method for preserving the symmetry of the $\mathbf{P}(k)$ matrix. The method is simple and it produces the desired results that allow the adaptive multivariate DFE to be stable. The issues regarding this divergence in this system can be further investigated in the near future. The focus now is the performance of the RLS algorithm compared to the LMS algorithm on the multivariate DFE.

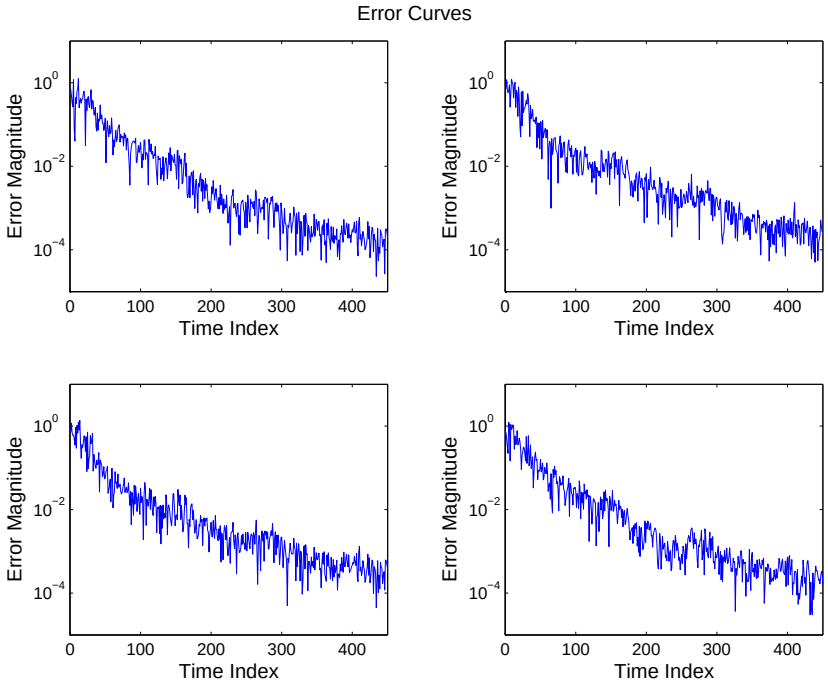

Figure 6.12 Effects of resetting the maximum value of the $\mathbf{P}(k)$ matrix on the error curve

6.2.5 Summary

In summary, the floating point simulation is an ideal environment in which to study system behaviour in the absence of fixed point issues and real-time aspects. The difficulty in implementing an adaptive multivariate DFE is far more complex in real-time. The difficulty include the effects of the real-time channel, hardware issues in the FPGA and getting the different modules to work successfully as an overall system.

The performance of the LMS and RLS algorithms are observed in the multivariate DFE. The algorithms perform relatively well in simulation, with the RLS algorithm having a performance advantage. From the error curve, the RLS algorithm converges much faster than the LMS algorithm. The decoded soft outputs produced by the RLS algorithm are far more defined and accurate compared to the LMS algorithm. The synchroniser that is implemented in the real-time system did not affect either algorithm in terms of these performance measures.

The simulations also show that the LMS and RLS algorithm both performed well in training mode. The synchroniser does not cause a problem during training mode and performs its duty for packet detection. The soft outputs produced by both algorithms are well-defined and the convergence rate of the error curves are reasonably fast. The RLS algorithm however has the issue of divergence in the $\mathbf{P}(k)$ matrix and notably resource consumption in the FPGA, discussed later in Chapter 7. In floating point, the divergence does not create a difficulty in terms of producing good results unless

saturation occurs as seen in Section 6.2.4. A method is devised to counter this problem, which is resetting the $\mathbf{P}(k)$ matrix to its initial value which prevents the eventual instability that comes from saturation. The next Section describes the simulation in fixed point using simulated and real-time received data. Once again, the adaptive algorithm performances are observed and the feasibility of the RLS algorithm is discussed.

6.3 FIXED POINT SIMULATION

6.3.1 Overview

The fixed point simulation reflects to a greater extent the actual performance produced by the FPGA on the Tait platform. Data is converted to 16-bit precision on the FPGA and this can be simulated as shown in this Section. The fixed point simulation uses both simulated and real-time received data. The real-time received data is extracted from the Tait platform which was described earlier in Chapter 4. Simulations were performed on the multivariate DFE with both adaptive algorithms. Their performance is illustrated and discussed below. In addition, the effects of the real-time system is briefly discussed.

The fixed point simulations are specifically set to 16-bit precision to test the Tait platform environment, based on its specifications seen in Table 4.1. The precision goes up to 32-bit for addition and subtraction, when two 16-bit numbers are multiplied. Numbers are scaled accordingly to 16-bit numbers if they are too large or too small and to avoid any fixed point effects, namely overflow. In hardware, overflow causes the number to saturate to its maximum. However, we know that saturation in the RLS algorithm leads to instability as seen in Section 6.2.4. In addition, it is well-known that the RLS algorithm can be sensitive to fixed point arithmetic [19]. Hence, it is important to see how it performs against the LMS algorithm in a fixed point environment.

The focus of this Section is the performance of these two algorithm in fixed point and how feasibile the RLS algorithm is for real-time implementation compared to the LMS algorithm. The Section also focusses particularly on the results directly relating to the Tait platform specifications.

6.3.2 Simulation Equalisation Results with a Quasi-Stationary Channel Model

This Section illustrates the performance of the adaptive filters in a fixed point environment using the quasi-stationary channel. The simulation is different from the floating point results, namely in the convergence rate of the algorithms. The convergence is much slower, most likely due to the conversion from floating point to fixed point. However, differences between the performances of the adaptive algorithms are still obvious.

Based on the floating point simulation, the prevention of intermediate variables from diverging is a key issue for implementation purposes. In general, the LMS algorithm did not exhibit any divergence due to its simplicity. However, for the RLS algorithm, the $\mathbf{P}(k)$ matrix did not converge except when the forgetting factor is set to 1. A simple mitigation technique has been proposed as seen in Section 6.2.4 and is applied in this fixed point simulation as seen in Section 6.3.3.

The results from the Section 6.2.2 suggests that a forgetting factor λ below 1 is not suitable for real-time implementation unless $\mathbf{P}(k)$ divergence can be dealt with which is discussed later. Hence, we consider a fixed point simulation with $\lambda = 1$. Specifically, the simulation parameters for these set of results are $\lambda = 1$, $\mu = 0.0625$, a tap length of 4, a training length of 1000 symbols and a packet length of 8000 symbols. The channel model is the quasi-stationary model mentioned in Section 4.6.2.

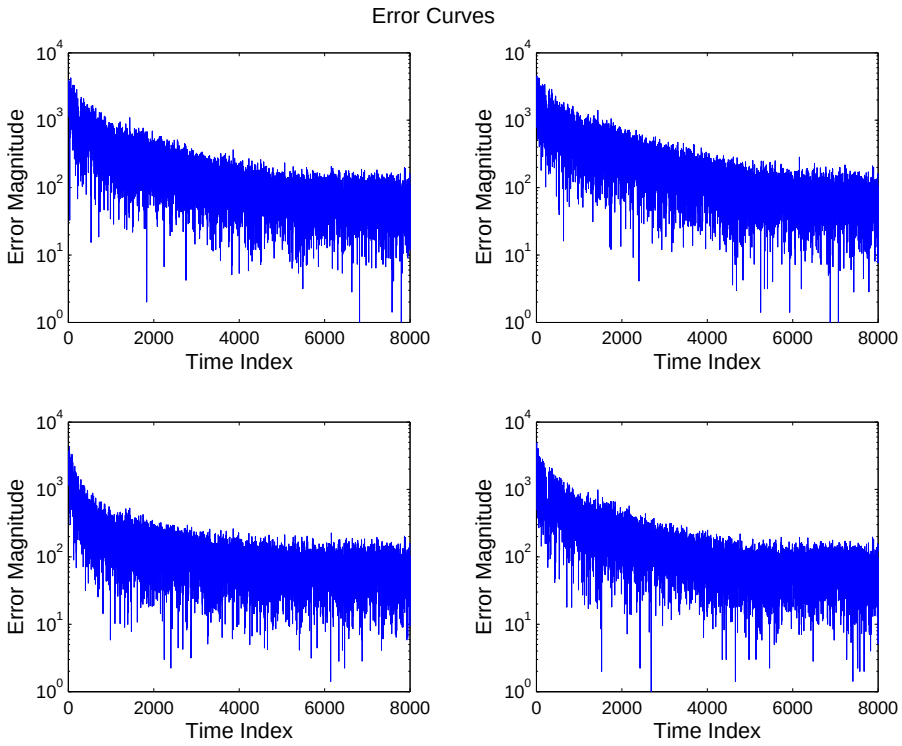

Figure 6.13 Error Curves for the LMS algorithm

Figures 6.13, 6.14, 6.15 and 6.16 show the outputs from the fixed point simulation. Evidently, after 1000 samples, the fixed point algorithms do not fully converge towards the error floor with the given training lengths as seen in Figures 6.13 and 6.14. Note that the actual values of the error magnitude are not comparable to the floating point simulation due to the fixed point structure. It is clear that the RLS algorithm converges faster than the LMS algorithm and reaches a lower error floor. Both algorithms produce reasonably accurate decoded soft outputs, with the RLS algorithm slightly more defined

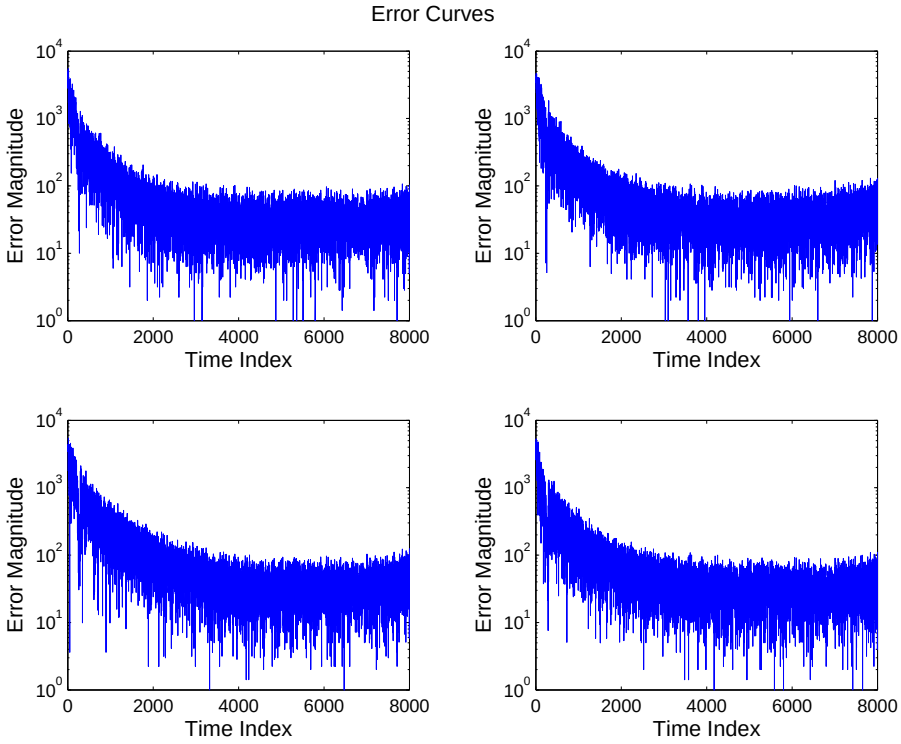

Figure 6.14 Error Curves for the RLS algorithm

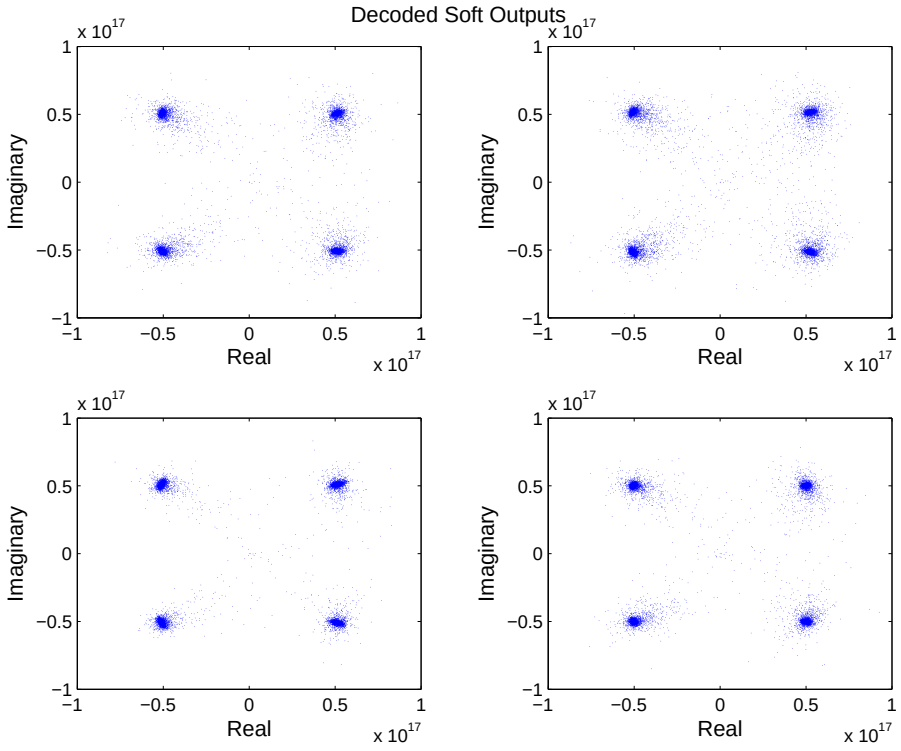

Figure 6.15 Decoded Soft Outputs for the LMS algorithm

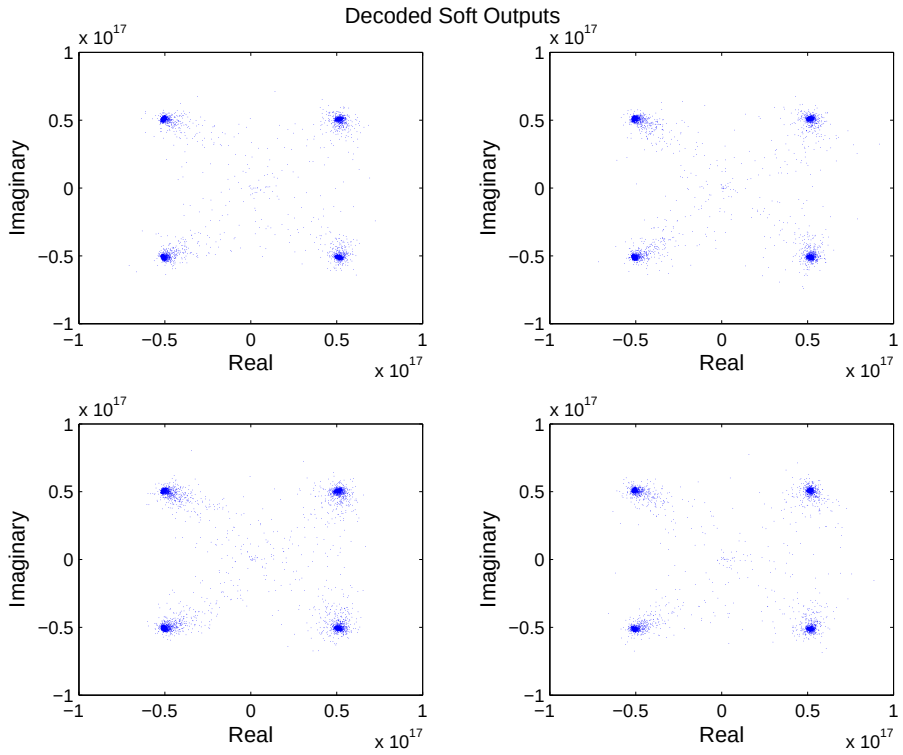

Figure 6.16 Decoded Soft Outputs for the RLS algorithm

as seen in Figures 6.15 and 6.16. Hence, for these parameters, the RLS algorithm does perform better than the LMS algorithm during the training mode and decision-directed mode.

6.3.3 Divergence Mitigation

In Section 6.2.4, we observed that saturation of the diverging $\mathbf{P}(k)$ matrix causes instability and that decision-delays did not mitigate this divergence for the RLS algorithm using $\lambda < 1$. This was also observed for the fixed point simulation using the quasi-stationary channel model. Basically, the results produced by $\lambda < 1$ caused diverging error curves and decoded soft outputs that were undesirable, therefore limiting the RLS algorithm to $\lambda = 1$ for fixed point and most likely its implementation on hardware. This is a major limitation of the RLS algorithm due to a single parameter setting, λ. Furthermore, setting $\lambda = 1$ defeats the purpose of the forgetting factor in putting more emphasis on recent data. Hence, its success in channels which vary is a cause for concern.

However, Section 6.2.4 suggested an alternative method which resets the $\mathbf{P}(k)$ matrix to its initial value whenever it diverges. Figures 6.17 and 6.18 show the results from this reset technique. The simulation parameters here are the same as in Section 6.3.2 except

that the forgetting factor is reduced to 0.99. Evidently, the technique stabilised the RLS algorithm when encountering saturation in a fixed point environment. Preservation of the symmetry as well as prevention of the increasing value of the $\mathbf{P}(k)$ matrix entries was observed. Hence, reinitialisation is seen to be a valid technique in stabilising the RLS algorithm in fixed point. The particular reset method used was to reset the matrix to its initial value when the maximum number allowed for 16-bits was reached (i.e. 2^{15}). Therefore, any real or imaginary part of the $\mathbf{P}(k)$ matrix entries that exceeded the maximum number of 2^{15} automatically forces the matrix to reset to its initial value. The results showed faster convergence in the error curves than $\lambda = 1$ and stability in decision-directed mode. The decoded soft outputs are even better than results with $\lambda = 1$.

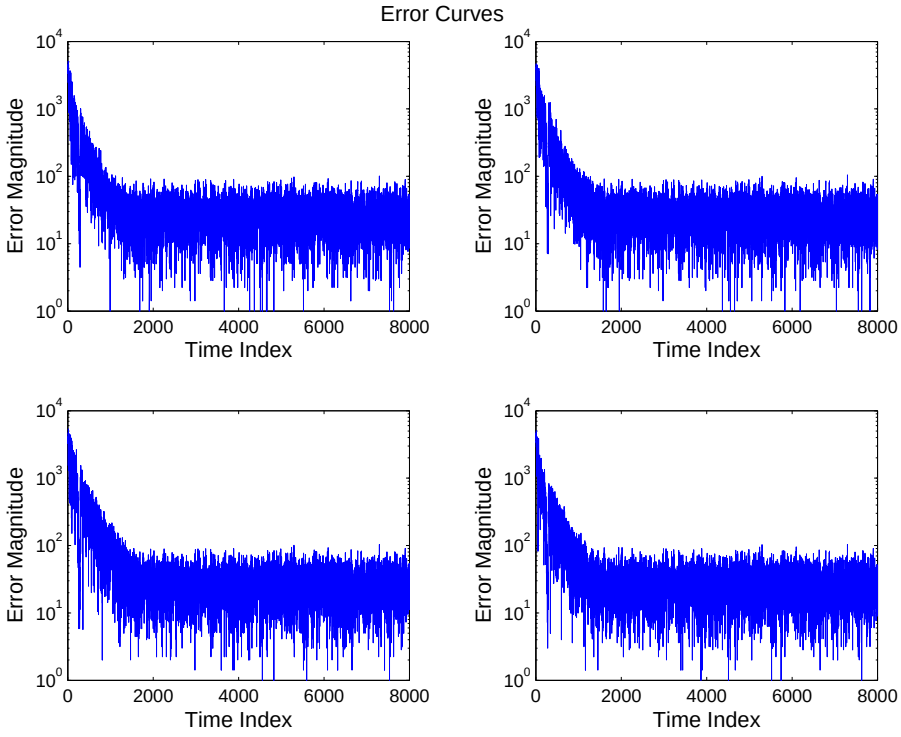

Figure 6.17 Error Curves for the RLS algorithm when resetting $\mathbf{P}(k)$

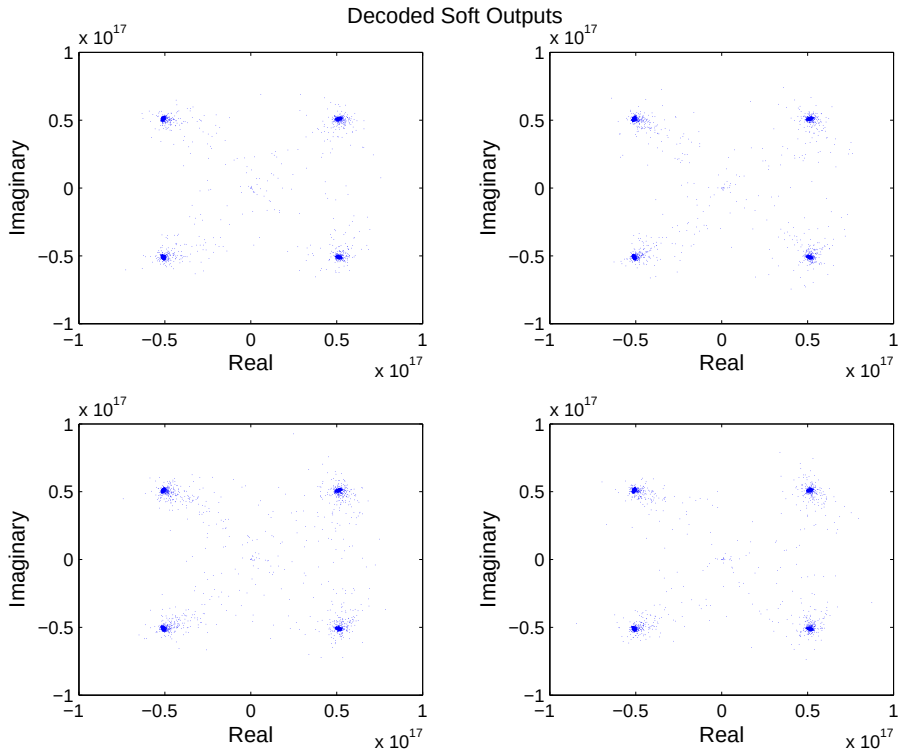

Figure 6.18 Decoded Soft Outputs for the RLS algorithm when resetting $\mathbf{P}(k)$

6.3.4 Simulation Equalisation Results with Captured Data

Overview

The previous Sections illustrated that both the LMS and RLS algorithms have been successfully implemented on the multivariate DFE. Both algorithms showed convergence in the error curves, with limitations on the RLS algorithm in the fixed point environment due to the diverging $\mathbf{P}(k)$ matrix. The mitigation of this divergence via resetting the $\mathbf{P}(k)$ matrix showed that the RLS algorithm can be stabilised but requires further study for implementation purposes. The focus now is the performance of the algorithms using real-time received data. The results shown in this Section use received data with 1000 training symbols transmitted at 0 dBm and a packet length of 8000 symbols. However, the buffer can only store up to 4000 symbols, thus results will show error curves up to 4000 symbols. The synchronisation period is located at the 256^{th} symbol, which can be observed in the error curves as a deep trough. Settings for the LMS algorithm are $\mu = 0.0625$ and the RLS algorithm uses $\lambda = 1$, $\Delta = 3$ and tap lengths are set at 4. The platforms are stationed approximately 20 metres apart, with the antennas facing away from each other. For comparison purposes, two sets of results are shown in this Section, representing two different sets of received data. In addition, results for the RLS algorithm with $\lambda < 0.99$ show that the algorithm does not work

here and exhibits instability. For tap lengths less than 3, the LMS and RLS algorithms do not work which coincides with the floating point and fixed point simulations shown earlier. Results for $\lambda = 1$ in the Tait simulations show better convergence than results using the Jakes model as seen in Chapter 5. It seems that for a quasi-stationary channel model, the RLS algorithm with an infinite memory is stable. However, any λ below 1 exhibits divergence.

First Example Set

This Section shows the error curves and decoded soft outputs for the first example set of received data. Figures 6.19 and 6.20 show the moving-average of the error curves for both algorithms in training mode. The two algorithms show different convergence properties than the floating and fixed point simulations, since real-time received data is used. The error curves show more variations in all 4 channels, particularly oscillations during decision-directed mode. Figures 6.23 and 6.24 show the decoded soft outputs for both algorithms, and for this particular example set, the RLS algorithm performs better, and all 4 received data produced desired results. For one channel, the decoded soft outputs for the LMS algorithm were not desirable. Figures 6.21 and 6.22 show the error curves over a longer period of time so that the decision-directed mode can be observed. These results show that the RLS algorithm exhibits pronounced oscillations during decision-directed mode.

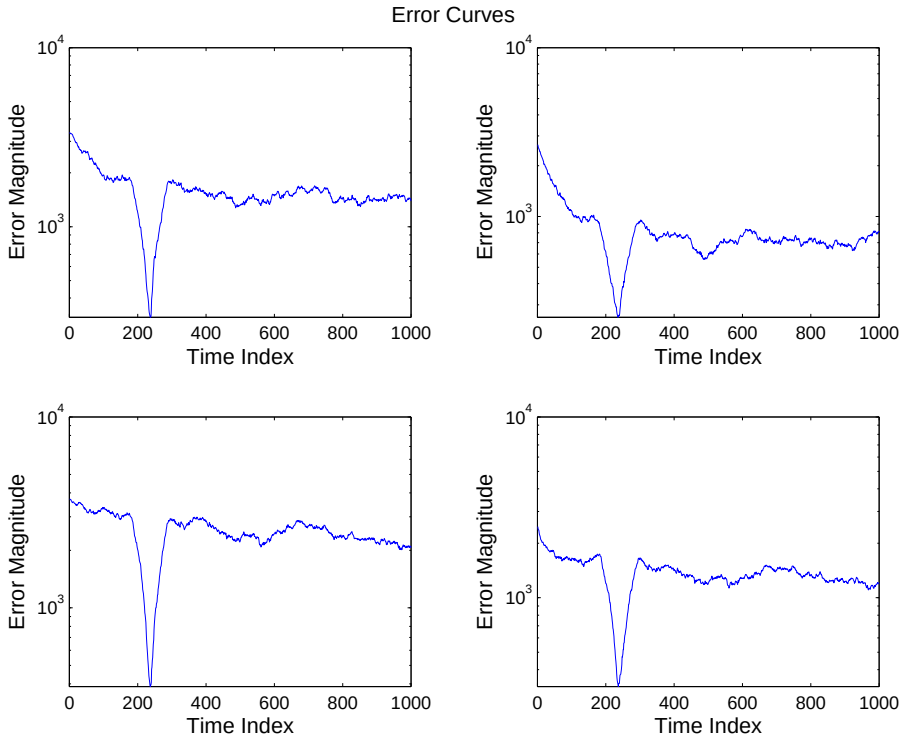

Figure 6.19 Moving-average of the error curves during training mode for the LMS algorithm using the first set of real-time received data

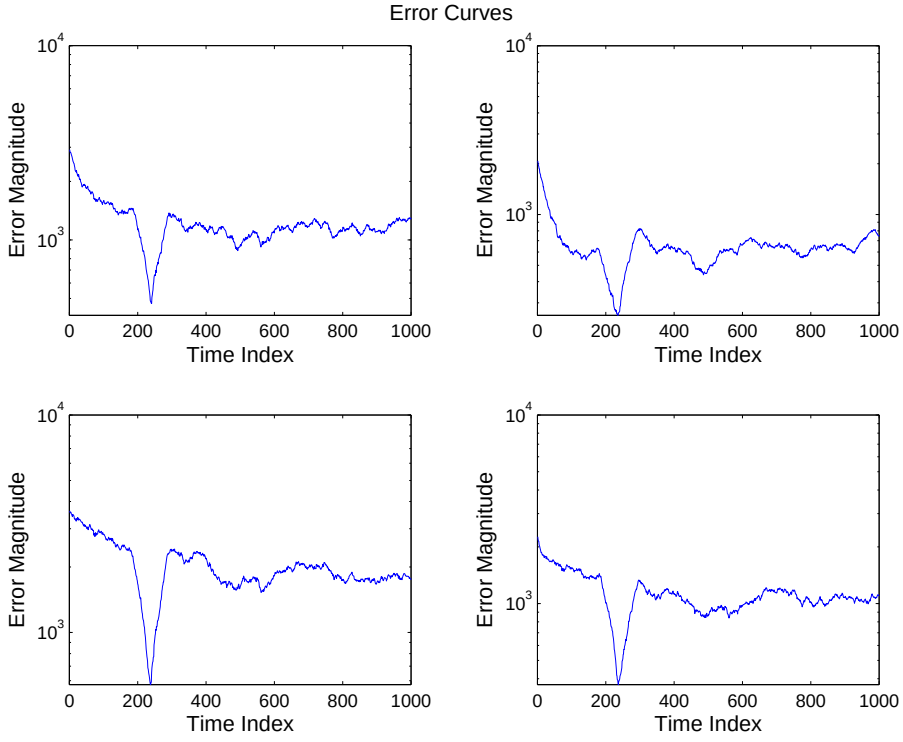

Figure 6.20 Moving-average of the error curves during training mode for the RLS algorithm using the first set of real-time received data

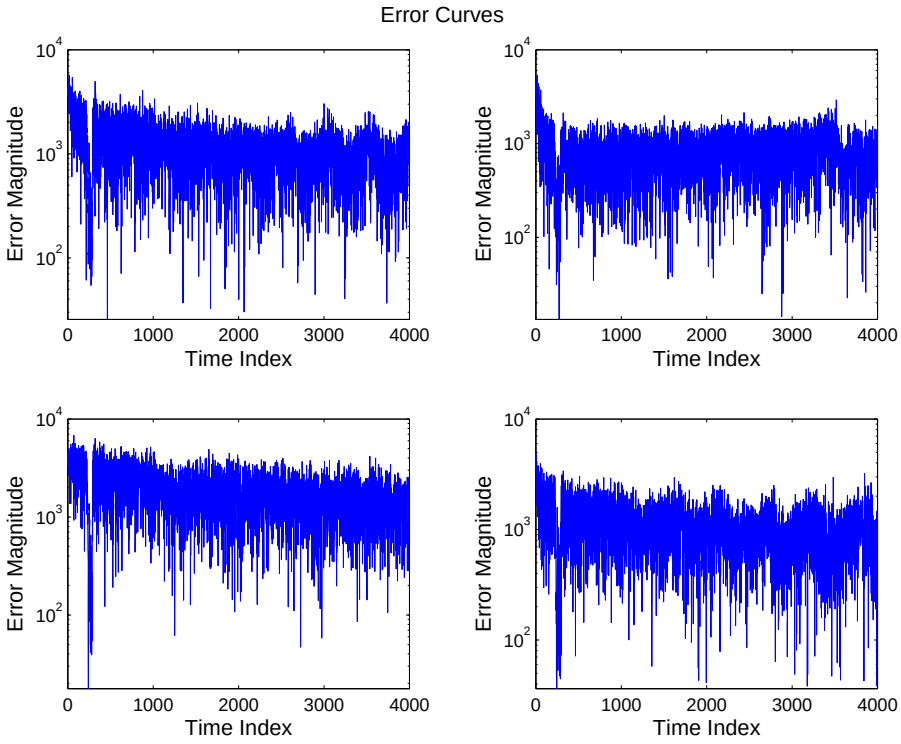

Figure 6.21 Error curves for the LMS algorithm using the first set of real-time received data

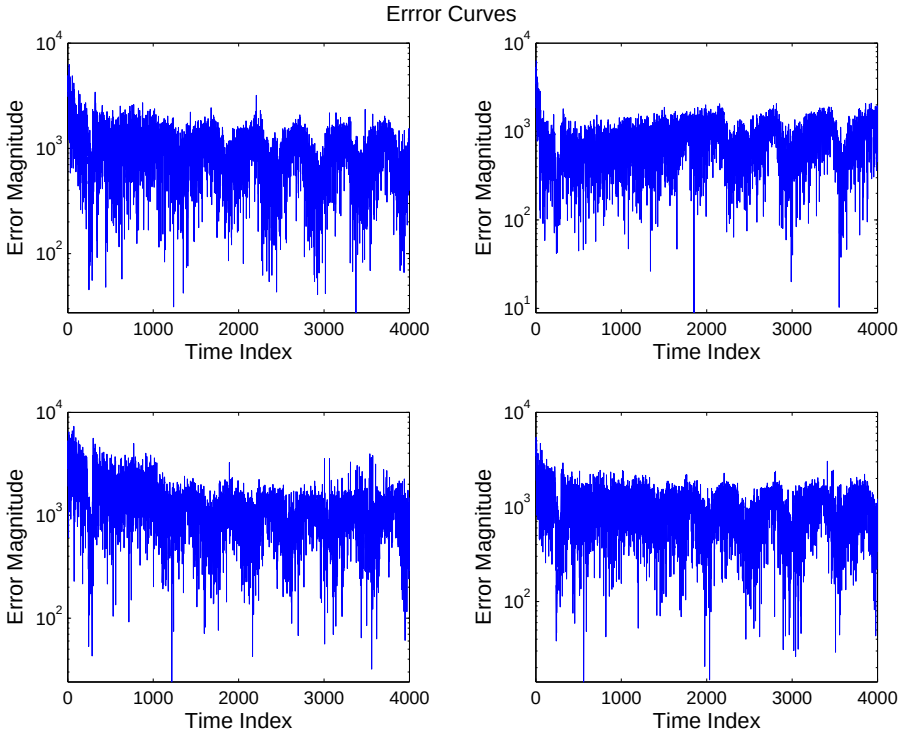

Figure 6.22 Error curves for the RLS algorithm using the first set of real-time received data

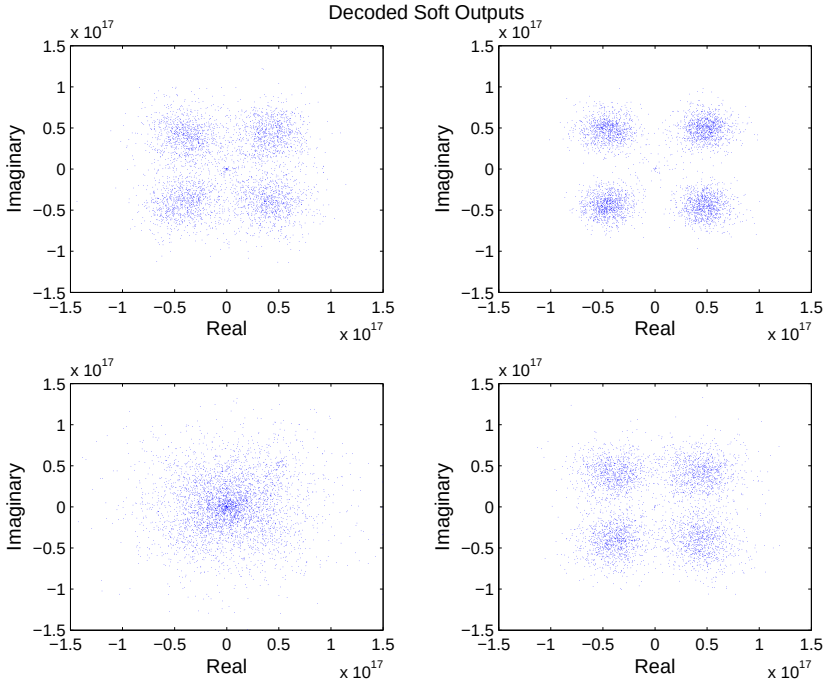

Figure 6.23 Decoded soft outputs for the LMS algorithm using the first set of real-time received data

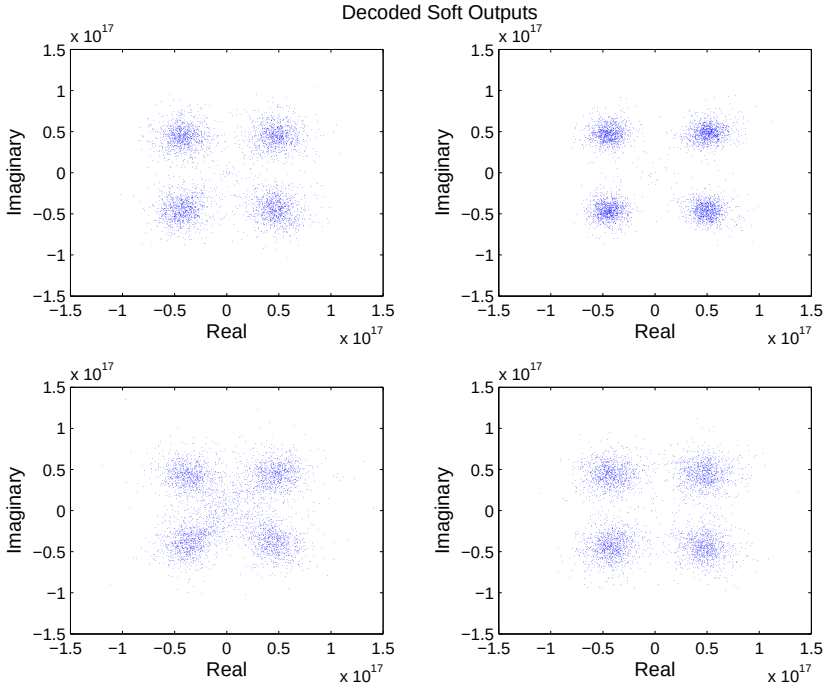

Figure 6.24 Decoded soft outputs for the RLS algorithm using the first set of real-time received data

Second Example Set

Similar to the previous Section, the error curves and decoded soft outputs for the second example set of received data are shown here. The moving-average error curves during training mode for this example set are similar to the previous set and therefore are not shown. Figures 6.25 and 6.26 show the error curves for both algorithms. The results show that both algorithms show convergence towards the error floor. However, the RLS algorithm showed less stability than the LMS algorithm during decision-directed mode, showing more fluctuations during tracking. Figures 6.27 and 6.28 show the decoded soft outputs for both algorithms. The results show that the decoded soft outputs for both algorithms are satisfactory producing the 4 distinct points. This example set shows the LMS algorithm performing better than the first example set, producing stable error curves and the desired decoded soft outputs. It also performs better than the RLS algorithm in terms of tracking because of fewer fluctuations during decision-directed mode.

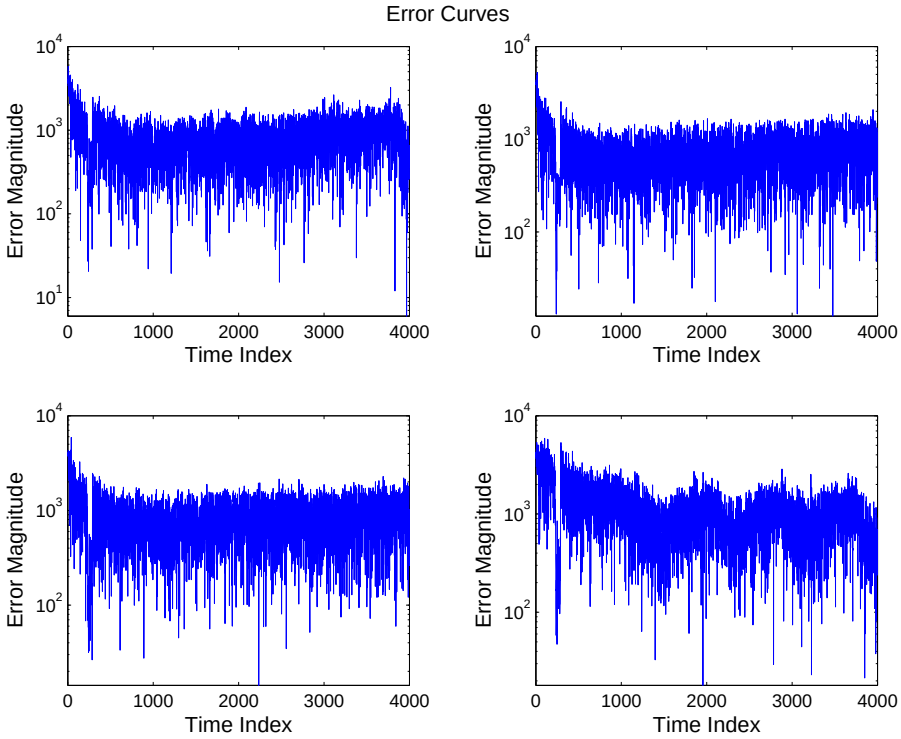

Figure 6.25 Error curves for the LMS algorithm using the second set of real-time received data

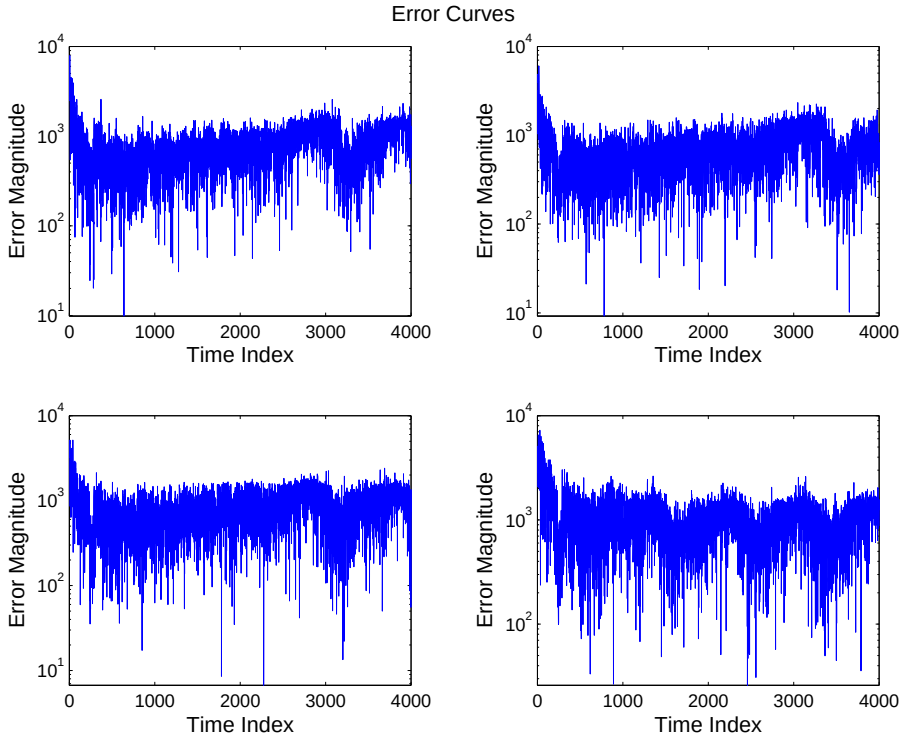

Figure 6.26 Error curves for the RLS algorithm using the second set of real-time received data

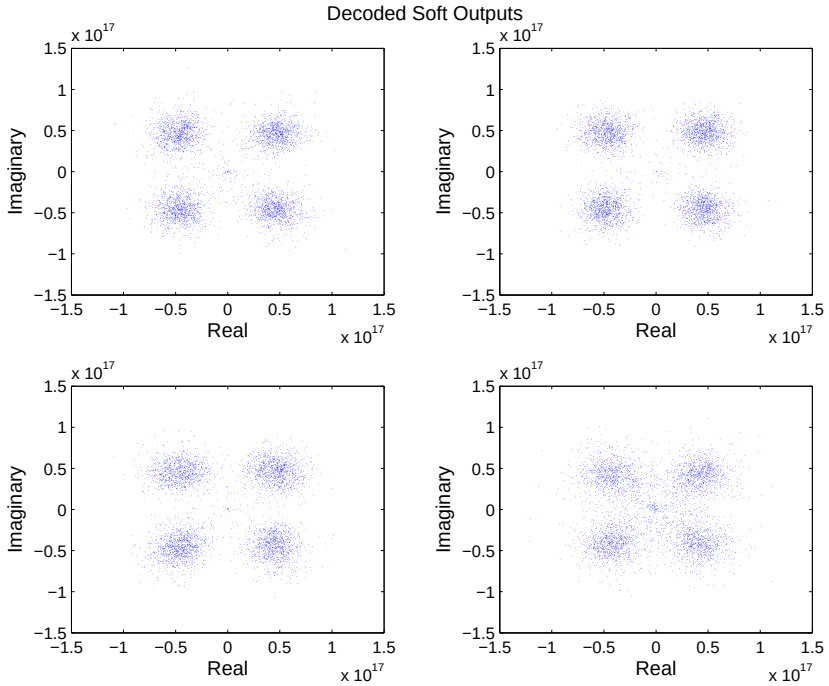

Figure 6.27 Decoded soft outputs for the LMS algorithm using the second set of real-time received data

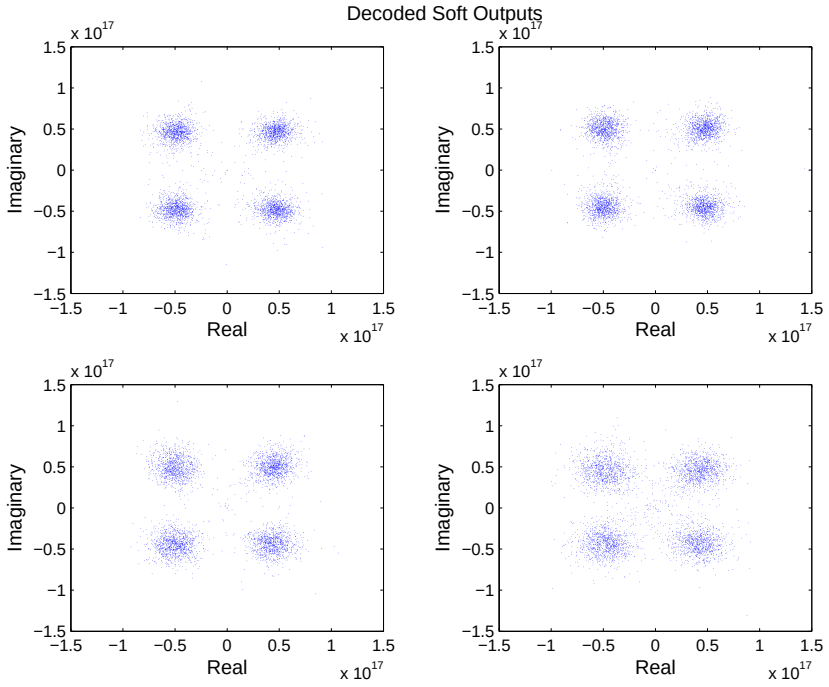

Figure 6.28 Decoded soft outputs for the RLS algorithm using the second set of real-time received data

Resetting $\mathbf{P}(k)$

Throughout this thesis, performance investigation of both adaptive algorithm shows that the RLS algorithm can outperform the LMS algorithm. The algorithm however is sensitive to parameter settings and, for example, when the forgetting factor λ is below 1, the matrix $\mathbf{P}(k)$ diverges. It was found that a decision-delay does not work in stabilising this matrix for the Tait simulations. However, resetting the matrix to its initial values does seem to work and this is seen in Section 6.2.4 and Section 6.3.3. Figures 6.29 and 6.30 show the two outputs from resetting the $\mathbf{P}(k)$ matrix whenever it diverges for a RLS algorithm using $\lambda = 0.99$. The matrix resets itself when it reaches the maximum number allowed in 16-bit precision (i.e. 2^{15}). This particular example uses the second set of received data as seen in Section 6.3.4. In general, the results show that the technique does work by stabilising the RLS algorithm, particularly during decision-directed mode except for one channel. By referring to Figure 6.29, the error curves produced by the second example set, resetting the matrix reduces the fluctuations during decision-directed mode. However, for one channel, the error curve which showed poor tracking as seen in Figure 6.29 resulted in instability, which inevitably produced undesirable decoded soft outputs as seen in Figure 6.30. In summary, the technique does work in stabilising the RLS algorithm and it seems to work best for channels that are relatively static.

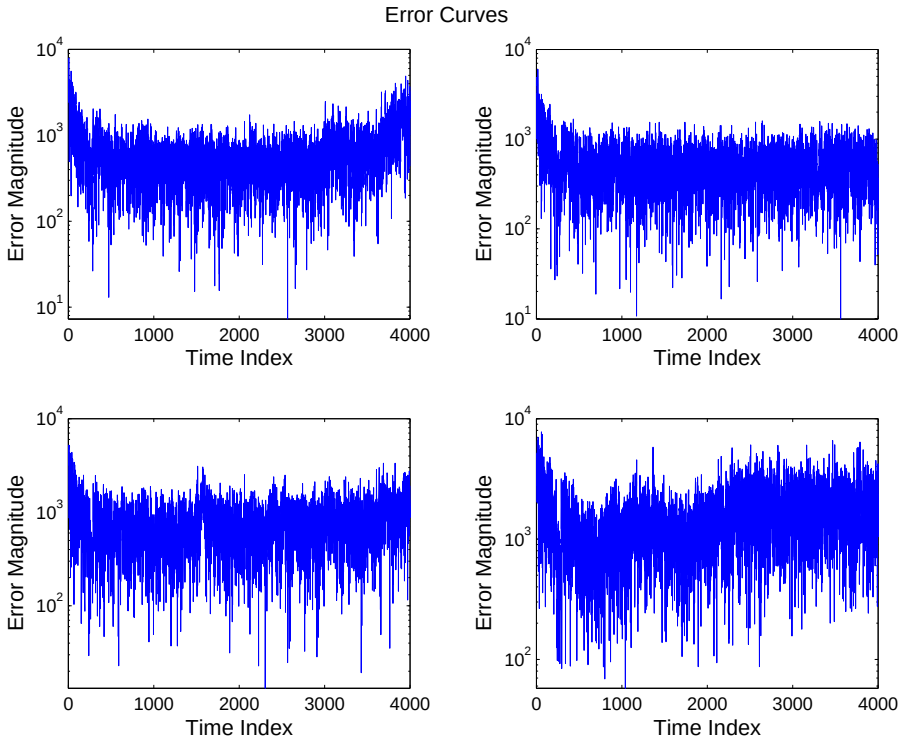

Figure 6.29 Error curves for the RLS algorithm using the reset technique

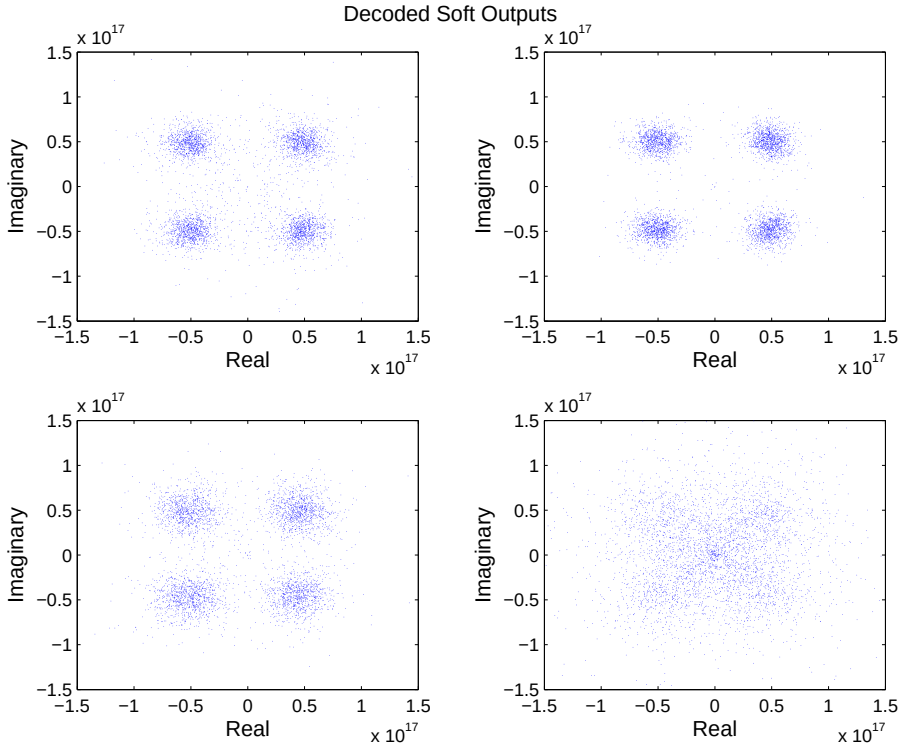

Figure 6.30 Decoded soft outputs for the RLS algorithm using the reset technique

Summary

In summary, these two particular sets of examples show that the LMS and RLS algorithms are similar in behaviour overall. There are some slight differences which can be observed in the error curves. The first example showed the RLS algorithm performing better than the LMS algorithm based on the soft decoded outputs. However, the first example showed that the RLS algorithm exhibited more oscillations in the decision-directed mode. The second example showed that the LMS algorithm performed as well as the RLS algorithm, and there was more stability in the LMS algorithm during decision-directed mode.

Basically, the results vary from one data set to another depending on the channel. The basic purpose of an adaptive algorithm is to counter this unknown real-time channel, because adaptive algorithms do not require a knowledge of the channel. The results show that both algorithms converge and produce relatively good decoded soft outputs.

In addition, the results for the RLS algorithm with $\lambda = 0.99$ did not converge. Applying the $\mathbf{P}(k)$ resetting technique helped mitigate the divergence. However, the results did not match the LMS algorithm. The technique does help mitigate instability, but the results for both set of examples show that the LMS algorithm performs as well as the RLS algorithm. It seems that mitigating the divergence of the RLS algorithm does not result in an approach which outperforms the LMS algorithm. Hence, due to the extra complexity, it is not worth the time and effort to implement on the Tait platform.

6.3.5 Discussion

Instability

In this Chapter, the floating and fixed point simulations showed that the RLS algorithm exhibited divergence in the $\mathbf{P}(k)$ matrix which leads to instability. Decision-delays did not help mitigate this, unlike results shown in Chapter 5 for the Jakes channel. The $\mathbf{P}(k)$ reset technique was introduced to mitigate this instability. Results show that the instability is indeed mitigated. However, performance of the RLS algorithm differs in floating and fixed point simulations. We observe that the effect of fixed point arithmetic is to reduce the superiority of the RLS algorithm compared to the LMS algorithm.

In addition, the RLS algorithm also showed instability during decision-directed mode in the fixed point simulations using real-time received data as seen in Section 6.3.4. The decision-directed mode showed oscillations more prevalent than in the LMS algorithm. This coincides well with [19] which states that the RLS algorithm is a poor tracking algorithm.

6.3.6 Real-time Issues

The fixed point simulation of the adaptive algorithms using real-time received data as seen in Section 6.3.4 showed that the convergence behaviour differs from that encountered using simulated received data. Although the precise causes of these differences is unknown, the outputs provided a good indication that the algorithms can work on the Tait platform and the RLS algorithm performs only marginally better.

6.3.7 Conclusion

In general, the adaptive algorithms have been succesfully simulated in floating and fixed point using the Tait specifications as seen in Table 4.1. A novel method was introduced, namely resetting the $\mathbf{P}(k)$ matrix for the RLS algorithm. Simulations were done in floating point and fixed point. In addition, real-time received data was used for simulations. The Chapter illustrated performance via error curves and decoded soft outputs which showed different behaviour as the simulation scenarios progressed.

The floating point simulations clearly showed the superiority of the RLS algorithm over the LMS algorithm, with the RLS algorithm converging much faster than the LMS algorithm. Howevever, the same issue was encountered as in the base simulation which was the effect of reducing λ to less than 1. This caused the $\mathbf{P}(k)$ matrix to diverge which is highly undesirable. A mitigation of this problem was found in Chapter 5 by implementing a decision-delay. This approach did not work in the Tait simulation, therefore a novel method was used. This method basically resets $\mathbf{P}(k)$ to its initial value and is a good method if an overflow or saturation occurs in the fixed-point simulation. The fixed-point simulation, on the other hand, produced very similar results for the LMS and RLS algorithms. The RLS algorithm exhibited more oscillations during decision-directed mode when using real-time received data and does not outperform the LMS algorithm.

In conclusion, the simulation of the adaptive algorithms showed that the RLS algorithm still performs better than the LMS algorithm in floating point environments. However, issues such as the diverging $\mathbf{P}(k)$, fixed-point effects and real-time issues with the Tait platform highly favours the simpler LMS algorithm due to its stability, ease of implementation and relatively good tracking during decision-directed mode.

Chapter 7

HARDWARE AND IMPLEMENTATION ISSUES

7.1 INTRODUCTION

The implementation of adaptive filter algorithms on the multivariate DFE on the FPGA is reasonably straightforward if there are no requirements for optimisation or complexity. However, this is not the case for the Tait platform which is limited by the size of its current FPGA, a Stratix EP1S25 described earlier in Chapter 4. It is known that the LMS algorithm implementation for the multivariate DFE on the Stratix EP1S25 has not fully consumed all the resources, notably half the DSP blocks are used. It is also worth nothing that other applications are implemented on the FPGA as well, namely the synchroniser. It is estimated that approximately 60% of the FPGA resources are already taken up by the LMS implementation. The RLS approach is clearly more complex and demands far more resources, as seen in Section 7.2. The next few Sections will describe the practical advantages of the LMS algorithm based on resource requirements, complexity and fixed point simulation results.

As we can see from the previous Chapters, the LMS and RLS algorithm have been studied. Their performances have been compared for several simulation scenarios. These simulation scenarios are summarised in Table 4.5. The results in Chapter 5 and 6 show that overall, the RLS algorithm outperforms the LMS algorithm in floating point environments. However, in fixed point environments the RLS algorithm only performs slightly better. It seems that the LMS algorithm is still more practical for implementation because it is less complex than the RLS algorithm. In addition, there are no instability issues with the LMS algorithm.

This Chapter will discuss hardware and implementation issues for the adaptive multivariate DFE. The estimated resource consumption by both the LMS and RLS algorithms are shown. This is followed by a feasibility discussion addressing issues like complexity, instability and performance of both algorithms.

7.2 RESOURCE REQUIREMENTS

It is difficult to accurately estimate the resource requirements for the LMS and RLS algorithm on the multivariate DFE without actual hardware implementation. However, it is imperative that resource requirements are discussed to a certain extent. In general, the resources in the FPGA can be summarised in the following categories below:

- Memory via RAM blocks
- Multipliers via DSP blocks
- Logic Elements

A rough estimate can be obtained to determine how complex the RLS algorithm is compared to the LMS algorithm for implementation. Logic elements are hard to determine because they can be reduced by optimising the structure and design of the VHDL code. Therefore, depending on design, logic elements cannot be determined until actual hardware implementation is performed. However, RAM and DSP block usage can be estimated.

7.2.1 RAM Blocks

Based on Chapter 4, memory availability is sufficient for the LMS algorithm on the multivariate DFE since it is already implemented on the Tait platform. For the RLS algorithm, it is easy to estimate the memory requirements by looking at the dominant variable that consumes memory, namely the $\mathbf{P}(k)$ matrix. Assuming the matrix size is 32×32, as described in Section 7.3.1, and the precision of each matrix entry is 16-bits, the configuration for memory storage is 1024×16. By referring to Section 4.5.5, the Stratix EP1S25 allows a maximum of two M-RAM blocks, and each M-RAM block allows a configuration of $32K \times 16$. Clearly, this is sufficient to store considerably more than a single $\mathbf{P}(k)$ matrix. One single $\mathbf{P}(k)$ consumes approximately 3% of the M-RAM block. Other variables that are stored include $[\ \mathbf{y}(k) \quad -\hat{\mathbf{x}}(k)\ ]$, $\mathbf{K}_p(k)$ and tap weights which involve minor additional memory requirements compared to $\mathbf{P}(k)$.

7.2.2 DSP Blocks

Based on Chapter 4, the multipliers are sufficient for the LMS algorithm on the multivariate DFE since it is already implemented on the Tait platform. Assuming that multiplications are done in 16-bit precision as done on the LMS algorithm implementation, the number of DSP blocks based on Section 4.5.4 allowed is 40. Referring to Section 4.7.3, a complex multiplier is equivalent to 4 real multipliers and therefore each clock cycle can only allow 10 complex multiplications. Clearly, to assess whether the number of multipliers is sufficient for RLS algorithm implementation, the number of

multiplications required needs to be taken into account. Table 7.1 shows the operations required for a single iteration for the RLS algorithm. The table summarises the operations in terms of the length of the vector L, where $L = mL_{ff} + nL_{fb}$. As defined before, L_{ff} and L_{fb} represents the tap length of the feedforward and feedback filter respectively, and n and m represents the number of transmit and receive antennas respectively. Section 3.4.4 shows these variables. Observing Table 7.1, the RLS

Operation	Size of Matrix Calculation	Adders	Multipliers
$\mathbf{P}(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H = tmp1$	$(L \times L)(L \times 1) = L \times 1$	$L^2 - 1$	L^2
$\begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} tmp1 + 1 = R_e(k)$	$(1 \times L)(L \times 1) + (1 \times 1) = (1 \times 1)$	L	L
$1/R_e(k) = R_e(k)^{-1}$	$(1 \times 1) = (1 \times 1)$	None	None
$tmp1 \times R_e(k)^{-1} = \mathbf{K}_p(k)$	$(L \times 1)(1 \times 1) = (L \times 1)$	None	L
$\begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H \mathbf{P}(k) = tmp2$	$(1 \times L)(L \times L) = (1 \times L)$	$L^2 - 1$	L^2
$\mathbf{K}_p(k) \times tmp2 = tmp3$	$(L \times 1)(1 \times L) = (L \times L)$	$L^2 - 1$	L^2
$\mathbf{P}(k) - tmp3 = \mathbf{P}(k+1)$	$(L \times L) - (L \times L) = (L \times L)$	L^2	None
$x_i(k) - \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{w}_i(k) = e_i(k)$	$(1 \times 1) - (1 \times L)(L \times 1) = (1 \times 1)$	nL	nL
$\mathbf{w}_i(k) + \mathbf{K}_p(k) e_i(k) = \mathbf{w}_i(k+1)$	$(L \times 1) + (L \times 1)(1 \times 1) = (L \times 1)$	nL	nL

Table 7.1 Table of Resource Requirements for the RLS algorithm, where $i = 1, \ldots, n$

algorithm requires $3L^2 + 2L + 2nL$ complex multipliers within a sample rate. Using specifications from Table 4.1, L is 32 and n is 4. Therefore, the number of complex multipliers required is 3392. Referring to Table 4.1, the system clock is 120MHz and the sample rate is 1MHz, which allows a maximum of 120 clock cycles. Therefore, since 10 complex multipliers are allowed within a clock cycle, the maximum number of multipliers that can be given within a sample rate is 1200 which is not sufficient for RLS algorithm implementation. It is also worth noting that this calculation does not incorporate λ which scales 3 of the variables above, $R_e(k)$, $\mathbf{K}_p(k)$ and $\mathbf{P}(k)$ as seen in Section 3.4.4. Therefore, this increases the resource requirements further, especially scaling $\mathbf{P}(k)$. This, in fact, would equate to $4L^2 + 3L + 2nL + 1$ complex multipliers.

Evidently, the LMS algorithm does not have intermediate variables such as $\mathbf{P}(k)$ and $\mathbf{K}_p(k)$ as seen in the RLS algorithm. It only consists of a weight update and filtering for every sample rate, thus drastically reducing the amount of multipliers required. Shown in Table 7.2 are the resource requirements for the LMS algorithm, which equates to $2nL + n$ complex multipliers. As mentioned previously, L is 32 and n is 4 for the Tait platform. This equates to a total of only 260 complex multipliers within 120 clock cycles, far less than the RLS algorithm.

Operation	Size of Matrix Calculation	Adders	Multipliers
$x_i(k) - \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix} \mathbf{w}_i(k)$ $= e_i(k)$	$(1 \times 1) - (1 \times L)(L \times 1)$ $= (1 \times 1)$	nL	nL
$\mathbf{w}_i(k) + \mu e_i(k) \begin{bmatrix} \mathbf{y}(k) & -\hat{\mathbf{x}}(k) \end{bmatrix}^H$ $= \mathbf{w}_i(k+1)$	$(L \times 1) + (1 \times 1)(1 \times 1)(L \times 1)$ $= (L \times 1)$	nL	$n(L+1)$

Table 7.2 Table of Resource Requirements for the LMS algorithm, where $i = 1, \ldots, n$

7.3 FEASIBILITY

7.3.1 Complexity

As seen in Section 7.2, the RLS algorithm is far more complex than the LMS algorithm. The RLS algorithm requires more multipliers and a greater memory usage compared to the LMS algorithm. This lead to the RLS algorithm requiring significantly more resources, making it less desirable for implementation.

One possibility for implementing the RLS algorithm is to reduce the tap length to the minimum feasible value of 3 (which is $L = 24$). However, from Table 7.1 this equates to 1944 complex multipliers which is still more than 1200. Another possibility is to reduce the precision of the multipliers from 18-bit to 9-bit, thus doubling the number of multipliers available. However, an 8-bit multiplication is low in precision and degrades the accuracy of the RLS algorithm. Therefore, these options are not considered. Another interesting concept is to look further into the $\mathbf{P}(k)$ matrix. For example, we may use only certain entries in the matrix and set the others to zero. This will reduce complexity and this is a possibility because of the $\mathbf{P}(k)$ reset technique. The reset technique resets all entries of the matrix to 0 except its main diagonal during the equalisation. Thus, the main diagonal entries may be the only entries of $\mathbf{P}(k)$ used for calculation. This can be considered in future work.

7.3.2 Fixed Point Simulation

It is important to note that the adaptive algorithms have been successfully implemented on the multivariate DFE. Furthermore, the adaptive algorithms have been simulated in a fixed point environment showing that these algorithms can work on the FPGA given conditions similar to those preset in the simulations. In addition, several issues for example, the effects of decision-delay, the forgetting factor and how it affects $\mathbf{P}(k)$ were encountered. These issues with the RLS algorithm do not seem to have been mentioned in literature [2, 38, 43, 44].

In general, the fixed point simulation is the closest thing to emulating the hardware implementation for the RLS algorithm on the FPGA. The system currently works in 16-bit precision for its multiplication. Its accumulate works in 32-bit precision. As

seen in Chapter 6, the simulations showed two different scenarios, a floating point and fixed point simulation. The floating point simulation evidently showed that the RLS algorithm outperformed the LMS algorithm. However, in fixed point, the RLS algorithm only performs slightly better than the LMS algorithm. The RLS algorithm also showed poor tracking ability during decision-directed mode compared to the LMS algorithm if the option to go with $\lambda = 1$ is taken when using real-time received data.

As mentioned many times in this thesis, when choosing $\lambda < 1$, the RLS algorithm can encounter instability. A mitigation of this instability by resetting $\mathbf{P}(k)$ showed that the RLS algorithm can still perform equalisation. The simulations showed that the RLS algorithm has limitations in terms of performance as well as sensitivity to parameter settings. Therefore, results from simulations, in addition to complexity issues, instability and resource requirement issues show that the RLS algorithm needs considerably more work before it can compete with the more robust LMS algorithm.

7.3.3 Discussion

The RLS algorithm is typically a better adaptive algorithm than the LMS algorithm, if resources are available and the implementation and parameter settings are tuned to the operating conditions. However, for this particular application to the multivariate DFE on the Tait platform, the LMS algorithm is more practical for implementation based on a complexity study and fixed point simulation results. Even though, the RLS algorithm may achieve slightly better results, there are also issues with instability as well as its poor tracking ability which eventually leads to poor performance. In addition, the RLS algorithm requires more resources in the FPGA than the LMS algorithm.

We can conclude, based on simulation results, that the performance advantage of the RLS algorithm is overshadowed by instability and complexity issues which were shown throughout this thesis. Feasibility of the RLS algorithm as a viable adaptive algorithm for implementation requires further work into mitigation of instability. Further work may include studying the $\mathbf{P}(k)$ matrix in detail for the RLS algorithm implementation. But in conclusion, the most feasible option at this point is to use the LMS algorithm, or consider variants of the LMS algorithm. There is one such variant of the LMS algorithm that performs as well as the RLS algorithm as seen in [28].

7.4 CONCLUSION

This Chapter discussed hardware and implementation issues regarding adaptive algorithm implementation on the multivariate DFE. Issues regarding resource requirements, complexity and feasibility of the LMS and RLS algorithms for implementation were discussed.

As mentioned before, the RLS algorithm is a better alternative to the LMS algorithm in terms of its performance. However, there were instability issues in the RLS algorithm that had to be mitigated. Evidently, there is a trade-off between performance, complexity and instability. The resource requirements by the RLS algorithm are significantly higher than the LMS algorithm. The feasibility of the RLS algorithm was briefly discussed with options for complexity reduction and a summary of the fixed point simulation results.

Overall, the LMS algorithm is more practical for implementation. For an alternative to be seriously considered for implementation, it has to be stable, insensitive to fixed-point arithmetic and outperform the LMS algorithm. In addition, it should be robust to parameter settings and the radio environment and not require too much extra complexity.

Chapter 8

CONCLUSION

8.1 DISCUSSION

The Tait platform is a reconfigurable hardware system for MIMO research. The platform currently has a 4 by 4 antenna arrangement and is expandable to a 12 by 12 system by utilising three interlinked 4 channel processing modules. The reconfigurability of the platforms allows changes in the system parameters, algorithms etc. The platform has successfully provided a real-time channel sounder, space-time coded systems and the current adaptive multivariate DFE system with the LMS algorithm [16, 17]. Adaptive equalisation is a key component in communication systems, because it greatly improves signal quality by mitigating ISI in the wireless channel. Therefore, a key part of this thesis covers adaptive equalisation in MIMO systems focussing on the performance of adaptive algorithms and their implementation on the multivariate DFE. In 2004, the Group Research department proposed a possible improvement to the platform, which was to implement the RLS algorithm on the multivariate DFE. This is because the RLS algorithm is the most common alternative to the LMS algorithm due to its fast converging nature and presumably its performance in time-varying channels. Even though work on the RLS algorithm on a SISO DFE is well-known, the RLS algorithm implementation on the multivariate DFE is fairly recent. Furthermore, there are many gaps in the literature, especially concerning implementation issues. As a result of the research in this thesis, the implementation of the RLS algorithm on multivariate DFE for the Tait platform has not been pursued. Basically, the RLS algorithm is not feasible for implementation because it is sensitive to parameter settings and is potentially unstable. Furthermore, results using the real time captured data show that the LMS and RLS algorithms have similar performance and the tracking behaviour of the RLS algorithm is poor compared to the LMS algorithm. Finally, the greater hardware requirements for the RLS algorithm on the FPGA, namely the Stratix EP1S25 makes the algorithm less desirable due to complexity requirements. The general conclusion is that the LMS algorithm is still the most practical algorithm for the Tait platform due to stability, ease of implementation and reasonable resource requirements.

In Chapter 5, the base simulations illustrated that the RLS algorithm has the potential to be a better alternative than the LMS algorithm. The fast-converging nature of the RLS algorithm was superior to the LMS algorithm. However, the simulation showed divergence in the $\mathbf{P}(k)$ matrix which is undesirable for implementation on hardware. Such behaviour in floating point, appears to have been overlooked in previous literature. The simulation showed this is easily mitigated by using appropriate decision delays for the DFE.

In Chapter 6, the Tait simulations illustrated that once again, the RLS algorithm outperformed the LMS algorithm in floating point, but exhibited the same diverging $\mathbf{P}(k)$ matrix. The simulations showed that the divergence in $\mathbf{P}(k)$ causes instability due to saturation. The decision delays did not mitigate the divergence in this case. A technique used to mitigate divergence was to reset the $\mathbf{P}(k)$ matrix to its initial value. This actually restored stability in the RLS algorithm [64, 63], presumably because it preserved the highly diagonal and symmetric nature of $\mathbf{P}(k)$. However, fixed point simulation using real time received data showed that the RLS algorithm suffers from tracking problems during decision-directed mode [19], and furthermore performs in a similar manner to the LMS algorithm for convergence.

In Chapter 7, hardware and implementation issues were discussed regarding the LMS and RLS algorithms. In summary, the complexity of the RLS algorithm is considerably more than the LMS algorithm. The resource requirements using the Stratix EP1S25 show that the RLS algorithm requires significantly more DSP blocks than the LMS algorithm on the Tait platform. Complexity reduction of the RLS algorithm is required for it to be considered for implementation on the Tait platform. However, in terms of feasibility, it seems that this alternative to the LMS algorithm is not suitable based on fixed point simulation results and stability issues. A more logical approach is to focus more on simulation work, rather than implementation on hardware.

Even though the RLS algorithm can provide better performance, as seen in the floating point simulations, the real-time implementation using Tait specifications actually produces slightly worse performance than the LMS due to fixed-point precision, real-time channel effects and hardware-related issues. Furthermore, there were stability issues encountered during simulations which hindered an implementation study of the RLS algorithm on hardware. With an existing system that produces the desired results, the LMS algorithm is clearly the better algorithm for implementation on the FPGA at present. The simplicity and stability of the LMS algorithm is far more desirable than the potential performance advantage offered by the RLS algorithm at the expense of sensitivity and instability. However, the RLS algorithm could still become useful in the future once all the instability issues are thoroughly resolved and the complexity requirements are satisfied. The obvious form for implementation is an RLS-LMS hybrid where RLS is used for convergence during training mode and LMS for tracking during decision-directed mode.

In addition to results from simulations, an initial study of the RLS algorithm on the multivariate DFE uncovered two issues worth noting. Firstly, the RLS algorithm suggested in [2] appears to have a typographical error as described in Section 3.5.2. Basically, when $\lambda < 1$, the RLS algorithm is unstable. Secondly, the adaptive multivariate DFE described in [2] did not incorporate decision delays, Δ, like other MIMO DFE work [38, 40, 18]. Results in Chapter 5 showed that the decision delays did not affect performance but did stabilise one of the RLS algorithm parameters, $\mathbf{P}(k)$. However, results in Chapter 6 showed that the decision delays did not stabilise the algorithm for the full Tait simulation, thus in some ways justifying the absence of the decision delay in [2].

There are several parameter combinations that have been discussed throughout the thesis. This makes it difficult to obtain an overview of the results. Hence, for clarification, a summary of the findings are shown in Table 8.1:

Parameter Settings			Jakes (float)	Quasi-stationary (float)	Quasi-stationary (fixed)	Captured-data (fixed)
$\lambda = 1$	$taps < 3$	$\Delta \leq taps - 1$	Y	X	X	X
		$\Delta > taps - 1$	Y	X	X	X
	$taps \geq 3$	$\Delta \leq taps - 1$	Y	Y	Y	Y
		$\Delta > taps - 1$	Y	Y	Y	Y
$\lambda < 1$	$taps < 3$	$\Delta \leq taps - 1$	Y	X	X	X
		$\Delta > taps - 1$	Y	X	X	X
	$taps \geq 3$	$\Delta \leq taps - 1$	O	O	X	X
		$\Delta > taps - 1$	Y	O	X	X

Table 8.1 Summary of RLS algorithm performance and parameter sensitivity for all the simulation scenarios, where Y = working, O = working but $\mathbf{P}(k)$ diverge, X = not working/unstable

Note that throughout this thesis, we have standardised the tap lengths such that the number of feedforward taps, L_{ff}, and feedback taps, L_{fb}, are the same. This common tap length is termed *taps* in Table 8.1. From Table 8.1 we can summarise the main points for the RLS implementation as below:

- All simulations with taps ≥ 3 show that for $\lambda = 1$, $\mathbf{P}(k)$ converges.
- The base simulations (using Jakes model) can give satisfactory results for any tap lengths.
- The base simulation with $\lambda < 1$ can cause $\mathbf{P}(k)$ to diverge but this can be fixed with appropriate decision delays.
- The Tait simulations (floating and fixed point) do not give satisfactory results for any tap lengths below 3.

- The Tait simulation with $\lambda < 1$ can cause $\mathbf{P}(k)$ to diverge and this cannot be fixed with appropriate decision delays.
- The Tait simulations (fixed point) with $\lambda < 1$ are sensitive to diverging $\mathbf{P}(k)$ because of saturation which affects the properties of $\mathbf{P}(k)$. This leads to instability.
- The Tait simulations (floating and fixed point) with $\lambda < 1$ can be stabilised by using the $\mathbf{P}(k)$ reset technique.

8.2 FUTURE WORK

Despite the extensive amount of work that has been done on adaptive equalisation, little has been done on real-time adaptive multivariate DFE implementation. The simulations in this thesis have successfully implemented the RLS algorithm on the multivariate DFE. In addition, the work confirmed that the RLS algorithm has the potential to be a fast-converging algorithm, outperforming the LMS algorithm. However, the RLS algorithm in the real time environment considered performs no better than the LMS algorithm. Clearly, the RLS algorithm has limitations that may be related to hardware specifications or the unknown channel. A further study of the causes of the reduced performance of the RLS algorithm is important.

One of the proposed goals of this thesis was to implement the RLS algorithm on the multivariate DFE which was reasonable considering that the LMS algorithm was successfully implemented on the Tait platform. However as mentioned many times, instability issues as well as limited resources on the FPGA prevented the implementation. It is logical not to pursue this path because more work is still required for the RLS algorithm. Furthermore, work on the adaptive multivariate DFE is still relatively new.

It is proposed that future work should include more simulation work on the multivariate DFE which may include using variants of the LMS algorithm, LMS-RLS algorithm hybrids and a more detailed investigation of the instability of the RLS algorithm.

8.3 SUMMARY

In general, this research gives system designers useful information about the viability of the LMS and RLS algorithms on a MIMO DFE. Rather than focus on the fast-converging RLS algorithm, the designer has to consider the surprisingly large number of problems with RLS that have been addressed in this thesis. These problems lie in the areas of sensitivity, instability and complexity. The work focusses in particular on the study of the RLS algorithm on the specific Tait platform. Hence, it may be difficult to generalise the results, but the concerns over the RLS algorithm are useful for any system designer to be aware of.

As each of the issues with the RLS algorithm was encountered, a solution was proposed and implemented. Although the end result performed no better than the LMS algorithm, the fact that these problems could be solved indicate that is it still possible for the benefits of the RLS algorithm to be attained. However, further research is required in this area.

REFERENCES

[1] J. K. Cavers, *Mobile channel characteristics*. Norwell: Kluwer Academic Publishers, 2000.

[2] S. Kuo, I. V. McLoughlin, J. Dowle, "A reconfigurable platform for MIMO Research - realtime implementation of 4×4 adaptive multi-variate DFE," *Australian Telecommunications, Networks and Applications Conference Proceedings*, vol. 20, no. 3, pp. 550–560, Jun. 2004.

[3] E. Telatar, "Capacity of multi-antenna Gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, Nov-Dec. 1995.

[4] J. Foschini, "Layered space-time architecture for wireless communications in a fading environment," *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41–59, 1996.

[5] M. Kessling, "Unifying analysis of ergodic MIMO capacity in correlated Rayleigh fading environments," *European Transactions on Telecommunications*, vol. 16, no. 1, pp. 17–35, Jan. 2005.

[6] S. N. Diggavi, N. Al-Dhahir, A. Stamoulis, A. R. Calderbank, "Great expectations: The value of spatial diversity in wireless networks," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 219–270, Feb. 2004.

[7] C. Dubuc, D. Starks, T. Creasy, H. Yong, "A MIMO-OFDM prototype for next-generation wireless WANs," *IEEE Communications Magazine*, vol. 42, no. 12, pp. 82–87, Dec. 2004.

[8] A. Adjoudani, E. C. Beck, A. P. Burg, G. M. Djuknic, T. G. Gvoth, D. Haessig, S. Manji, M. A. Milbrodt, M. Rupp, D. Samardzija, A. B. Siegel, T. Sizer, C. Tran, S. Walker, S. A. Wilkus, P. W. Wolniansky, "Prototype experience for MIMO BLAST over third-generation wireless system," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 440–451, Apr. 2003.

[9] J. P. Kermoal, L. Schumacher, K. I. Pedersen, P. E. Mogensen, F. Frederiksen, "A stochastic MIMO radio channel model with experimental validation," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 6, pp. 1211–1226, Aug. 2002.

[10] J. W. Wallace, M. A. Jensen, "Modeling the indoor MIMO wireless channel," *IEEE Transactions on Antennas and Propagation*, vol. 50, no. 5, pp. 591–599, May. 2002.

[11] E. Bonek, H. Ozcelik, M. Herdin, W. Weichselberger, J. Wallace, "Deficiencies of a popular stochastic MIMO radio channel model," *Electronic Letters*, vol. 39, no. 16, pp. 1209–1210, Aug. 2003.

[12] P. Soma, D. S. Baum, V. Erceg, R. Krishnamoorthy, A. J. Paulraj, "Analysis and modeling of multiple-input multiple-output (MIMO) radio channel based on outdoor measurements conducted at 2.5 ghz for fixed BWA applications," *IEEE International Conference on Communications*, vol. 1, pp. 272–276, Apr. 2002.

[13] C. C. Martin, J. H. Winters, N. R. Sollenberger, "Multiple-input multiple-output (MIMO) radio channel measurements," *IEEE Antennas and Propagation Society International Symposium*, vol. 1, pp. 418–421, Jul. 2001.

[14] D. Chizhik, J. Ling, P. W. Wolniansky, R. A. Valenzuela, N. Costa, K. Huber, "Multiple-input-multiple-output measurements and modeling in Manhattan," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 321–331, Apr. 2003.

[15] I. V. McLoughlin, T. Scott, "Space-time processing Linux style," *Linux Journal*, no. 125, Sep. 2004.

[16] S. Kuo, I. V. McLoughlin, K. Mehrotra, "Reconfigurable processing framework space-time block codes," *Australian Telecommunications, Networks and Applications Conference*, Dec. 2003.

[17] K. Mehrotra, I. V. McLoughlin, "Time reversal space time block coding with channel estimation and synchronisation errors," *Australian Telecommunications, Networks and Applications Conference*, Dec. 2003.

[18] C. Tidestav, "The multivariable decision feedback equaliser," Ph.D. dissertation, Uppsala University, 1999.

[19] S. Haykin, *Adaptive filter theory.* Upper Saddle River, N.J.: Prentice-Hall, Inc., 1999.

[20] ——, *Introduction to adaptive filters.* Upper Saddle River, N.J.: Prentice-Hall, Inc., 1984.

[21] A. H. Sayed, *Fundamentals of adaptive filtering.* New York: IEEE Press Wiley-Interscience, 2003.

[22] M. E. Austin, "Decision-feedback equalization for digital communications over dispersive channels," *MIT Research Laboratory of Electronics Technical Report*, no. 473, Aug. 1967.

[23] B. Widrow, M. E. Hoff, "Adaptive switching circuits," *IRE WESCON Convention Record*, pp. 96–104, 1960.

[24] D. N. Godard, "Channel equalization using a Kalman filter for fast data transmission," *IBM Journal of Research and Development*, vol. 18, no. 3, pp. 267–273, May. 1974.

[25] D. D. Falconer, L. Ljung, "Application of fast Kalman estimation to adaptive equalization," *IEEE Transactions on Communications*, vol. 26, no. 10, pp. 1439–1446, Oct. 1978.

[26] W. Zhuang, "RLS algorithm with variable forgetting factor for decision feedback equalizer over time-variant fading channels," *Wireless Personal Communications: An International Journal*, vol. 8, no. 1, pp. 15–29, Aug. 1998.

[27] S. Nagaraj, S. Gollamudi, S. Kapoor, Y. Huang, J. R. Deller, Jr., "Multiuser detection based on a deterministic error specification: Theory and low-complexity adaptive solutions," *Asilomar Conference on Signals, Systems and Computers*, vol. 1, no. 31, pp. 801–805, Nov. 1997.

[28] Y. Lau, Z. M. Hussain, R. Harris, "Performance of adaptive filtering algorithms: A comparative study," *Australian Telecommunications Networks and Applications Conference*, Dec. 2003.

[29] S. U. H. Qureshi, "Adaptive equalization," *IEE Proceedings*, vol. 73, no. 9, pp. 1349–1387, Sep. 1985.

[30] B. Holter, "Multiple input-multiple output (MIMO) receiver for wideband space-time communications," *IEEE Norwegian Symposium on Signal Processing*, pp. 167–172, Oct. 2001.

[31] N. Al-Dhahir, A. H. Sayed, "The finite-length multi-input multi-output MMSE-DFE," *IEEE Transactions on Signal Processing*, vol. 48, no. 10, pp. 2921–2936, Oct. 2000.

[32] A. Maleki-Tehrani, B. Hassibi, J. Cioffi, "An estimation-based approach to multiple-input multiple-output (MIMO) channel equalization," *IEEE Sensory Array and Multichannel Signal Processing Workshop*, pp. 27–31, Mar. 2000.

[33] K. Suh, O. K. Ersoy, "Multivariable decision feedback equalizer for space diversity in multi-input/output channel," *IEEE Vehicular Technology Conference*, vol. 1, no. 55, pp. 31–34, May. 2002.

[34] Z. Yang, H. Zhu, "A comparison of interference cancellation and multiuser detection," *Asia-Pacific Conference on Enviromental Electromagnetics*, pp. 76–81, May. 2000.

[35] J. Louveaux, L. Vandendorpe, L. Cuvelier, F. Deryck, O. van de Wiel, "Linear and decision-feedback MIMO equalization for transmultiplexer-based high bit rate transmission over copper wires," *International Zurich Seminar on Broadband Communications*, pp. 177–184, Feb. 1998.

[36] R. Sebastian, D. D. Falconer, "Multi-user decision-feedback space-time equalization and diversity reception," *Vehicular Technolgy Conference*, vol. 1, no. 49, pp. 494–498, May 1999.

[37] J. Frigon, B. Daneshrad, "Multiple input-multiple output (MIMO) receiver for wideband space-time communications," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, no. 12, pp. A164–A168, Sept-Oct. 2001.

[38] ——, "A multiple input-multiple output (MIMO) adaptive decision feedback equalizer (DFE) with cancellation for wideband space-time communications," *International Journal of Wireless Information Networks*, vol. 9, no. 1, pp. 13–23, Jan. 2002.

[39] C. Komninakis, C. Fragouli, A. H. Sayed, R. D. Wessel, "Adaptive multi-input multi-output fading channel equalization using Kalman estimation," *IEEE International Conference on Communications*, vol. 3, pp. 1655–1659, Jun. 2000.

[40] ——, "Multi-input multi-output fading channel tracking and equalization using Kalman estimation," *IEEE Transactions on Signal Processing*, vol. 50, no. 5, pp. 1065–1076, May. 2002.

[41] L. Vandendorpe, O. van de Wiel, "MIMO DFE equalization for multitone DS/SS systems over multipath channels," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 3, pp. 502–511, Apr. 1996.

[42] R. Wang, N. Jindal, T. Bruns, A. R. S. Bahai, D. C. Cox, "Comparing RLS and LMS adaptive equalizers for nonstationary wireless channels in mobile ad hoc networks," *IEEE International Symposium on personal, Indoor and Mobile Radio Communications*, vol. 3, no. 13, pp. 1131–1135, Sep. 2002.

[43] A. Maleki-Tehrani, B. Hassibi, J. Cioffi, "Adaptive equalization of multiple-input multiple-output frequency selective channels," *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems and Computers*, vol. 1, no. 33, pp. 547–551, Oct. 1999.

[44] ——, "Adaptive equalization of multiple-input multiple-output (MIMO) channels," *IEEE International Conference on Communications*, vol. 3, pp. 1670–1674, Jun. 2000.

[45] B. C. Banister, J. R. Zeidler, "Tracking performance of the RLS algorithm applied to an antenna array in a realistic fading environment," *IEEE Transactions On Signal Processing*, vol. 50, no. 5, pp. 1037–1050, May. 2002.

[46] S. Ciochina, C. Paleologu, A. A. Enescu, "On the behaviour of RLS adaptive algorithm in fixed-point implementation," *International Symposium on Signals, Circuits and Systems*, vol. 1, pp. 165–168, Jul. 2003.

[47] M. Kocic, D. Brady, "Real-time RLS-MLSE equalizer implementation for application to PDC (personal digital cellular)," *International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2691–2694, May. 1995.

[48] M. M. Mokhtar, A. S. Ragab, "Performance analysis of adaptive DFE-RLS in mobile fading channels," *Proceedings of the Thirteenth National Radio Science Conference*, pp. 555–562, Mar. 1996.

[49] C. Drewes, R. Hasholzner, J. S. Hammerschmidt, "Adaptive equalization for wireless ATM," *International Conference on Digital Signal Processing*, no. 13, 1997.

[50] T. Lin, "Adaptive equalisation and mobile radio system," Master's thesis, University of Canterbury, 1999.

[51] A. P. Liavas, P. A. Regalia, "On the numerical stability and accuracy of the conventional recursive least squares algorithm," *IEEE Transactions in Signal Processing*, vol. 47, no. 1, pp. 88–96, Jan. 1999.

[52] M. Bouchard, "Recursive least-squares algorithms with improved numerical stability and constrained least-squares algorithms for multi-channel active noise control system," *Journal of the Canadian Acoustical Association*, vol. 28, no. 3, pp. 66–67, Sep. 2002.

[53] R. Zukunft, S. Haar, T. Magesacher, "A blind adaptation algorithm for decision feedback equalization for dual-mode CAP-QAM reception," *IEEE Global Telecommunications Conference*, vol. 1, pp. 307–311, Nov. 2002.

[54] A. Bouttier, "A truly recursive blind equalization algorithm," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 6, pp. 3381–3384, May. 1998.

[55] Z. Tian, "Mitigating error propagation in decision-feedback equalization for multiuser CDMA," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 525–529, Apr. 2004.

[56] J. Tsimbinos, L. B. White, "Error propagation and recovery in decision-feedback equalizers for nonlinear channels," *IEEE Transactions on Communications*, vol. 49, no. 2, pp. 239–242, Feb. 2001.

[57] R. Lopez-Valcarce, "Realizable linear and decision-feedback equalizers: Properties and connections," *IEEE Transactions on Signal Processing*, vol. 52, no. 3, pp. 757–773, Mar. 2004.

[58] S. Perreau, L. B. White, P. Duhamel, "A blind decision feedback equalizer incorporating fixed lag smoothing," *IEEE Transactions on Signal Processing Letters*, vol. 48, no. 5, pp. 1315–1328, May. 2000.

[59] W. J. Hillery, M. D. Zoltowski, M. Fimoff, "Decision feedback equalizer design for insensitivity to decision delay," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 505–508, Apr. 2003.

[60] N. Al-Dhahir, J. M. Cioffi, "Efficient computation of the delay-optimized finite-length MMSE-DFE," *IEEE Transactions on Signal Processing*, vol. 44, no. 5, pp. 1288–1292, May. 1996.

[61] ——, "MMSE decision-feedback equalizers: Finite-length results," *IEEE Transactions on Information Theory*, vol. 41, no. 4, pp. 961–975, Jul. 1995.

[62] C. Komninakis, C. Fragouli, A. H. Sayed, R. D. Wessel, "Channel estimation and equalization in fading," *Asilomar Conference on Signals, Systems, and Computers*, vol. 2, no. 33, pp. 1159–1163, Oct. 1999.

[63] M. E. Salgado, G. C. Goodwin, and H. M. Richard, "Modified least squares algorithm incorporating exponential resetting and forgetting," *International Journal of Control*, vol. 47, no. 2, pp. 477–491, 1988.

[64] A. Vahidi, A. Stefanopoulou, H. Peng, "Recursive least squares with forgetting for online estimation of vehicle mass and road grade: Theory and experiments," *Journal of Vehicle System Dynamics*, vol. 43, pp. 31–57, 2005.

[65] S. Haykin, *Communication systems.* New York: John Wiley and Sons, Inc., 1999.

[66] J. G. Proakis, *Digital communications.* New York: McGraw-Hill, 1989.

[67] M. H. Hayes, *Statistical digital signal processing and modelling.* New York: John Wiley and Sons, Inc., 1996.

[68] T. S. Rappaport, *Wireless communications: principles and practice.* Upper Saddle River, N.J: Prentice Hall, 2002.

[69] T. M. Cover, J. A. Thomas, *Elements of information theory.* New York: John Wiley and Sons, Inc., 1991.

[70] G. G. Raleigh, J. M. Cioffi, "Spatio-temporal coding for wireless communications," *IEEE Transactions on Communications*, vol. 3, no. 3, pp. 357–366, Mar. 1996.

[71] D. Gesbert, M. Shafi, S. Da-shan, P. J. Smith, A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 281–302, Apr. 2003.

[72] A. Tulino, S. Verdu, "Random matrix theory and wireless communications," *Foundations and Trends in Communications and Information Theory*, Feb. 2004.

[73] M. Shafi, D. Gesbert, S. Da-shan, P. J. Smith, W. H. Tranter,, "Guest editorial MIMO systems and applications I," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 277–280, Apr. 2003.

[74] ——, "Guest editorial MIMO systems and applications II," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 5, pp. 681–683, Jun. 2003.

[75] D. Tse, P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge: Cambridge University Press, 2005.

[76] A. Goldsmith, *Wireless Communications*. Cambridge: Cambridge University Press, 2005.

[77] A. Paulraj, R. Nabar, D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge: Cambridge University Press, 2003.

[78] G. E. Bottomley, S. T. Alexander, "A theoretical basis for divergence of conventional recursive least squares filters," *International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 908–911, May. 1989.

[79] J. Coon, M. Beach, J. McGeehan, M. Sandell, "Channel and noise variance estimation and tracking algorithms for unique-word based single-carrier system," *International Symposium on Wireless Communication Systems*, no. 1, pp. 51–55, Sep. 2004.

[80] C. J. Min, J. S. Kim, J. H. Park, J. W. Chong, "Fast Kalman/LMS algorithms on the strong multipath channel," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. II–465–II–468, Apr. 2003.

[81] Y. Gong, C. F. N. Cowan, "Optimum decision delay of the finite-length DFE," *IEEE Signal Processing Letters*, vol. 11, no. 11, pp. 858–861, Nov. 2004.

[82] Altera Stratix, *Stratix Device Handbook*. San Jose: Altera Corporation, 2004.

[83] W. C. Jakes Jr., *Microwave mobile communications*. New York: John Wiley and Sons, Inc., 1974.

[84] Y. Li, X. Huang, "The generation of independent Rayleigh faders," *IEEE Transactions on Communications*, vol. 1, pp. 41–45, Jun. 2000.

[85] ——, "The simulation of independent Rayleigh faders," *IEEE Transactions on Communications*, vol. 50, no. 9, pp. 1503–1514, Sep. 2002.

[86] Z. Wu, "Model of independent Rayleigh faders," *IEEE Electronic Letters*, vol. 40, no. 15, pp. 949–951, Jul. 2004.